A Benchmark Suite for Verifying Neural Anomaly Detectors in Distillation Processes

Simon Lutz[1], Justus Arweiler[2], Aparna Muraleedharan[3], Niklas Kahlhoff[4], Fabian Hartung[5,6], Indra Jungjohann[2], Mayank Nagda[6], Daniel Reinhardt[7], Dennis Wagner[6], Jennifer Werner[8], Justus Will[9], Jakob Burger[3], Michael Bortz[8] Hans Hasse[2], Sophie Fellenz[6], Fabian Jirasek[2], Marius Kloft[6], Heike Leitte[7], Stephan Mandt[9], Steffen Reithermann[6], Jochen Schmid[8], and Daniel Neider[10]

[1] Member of the Research Center "Trustworthy Data Science and Security" of the University Alliance Ruhr, TU Dortmund University, Germany
[2] Laboratory of Engineering Thermodynamics, University of Kaiserslautern-Landau
[3] Chemical Process Engineering, Technical University of Munich
[4] TU Dortmund University
[5] Gas Treatment Technology, BASF SE Ludwigshafen
[6] Machine Learning Group, University of Kaiserslautern-Landau
[7] Visual Information Analysis, University of Kaiserslautern-Landau
[8] Fraunhofer Institute for Industrial Mathematics ITWM
[9] University of California, Irvine
[10] Professor for Verification and Formal Guarantees of Machine Learning, Research Center "Trustworthy Data Science and Security" of the University Alliance Ruhr, TU Dortmund University, Germany

Abstract. Inspired by the success of machine learning in other domains, the application of AI in the field of chemical process engineering has increased in recent years. While neural networks often show paramount performance, they are error-prone in general which is particularly problematic when employed in safety-critical applications, such as chemical plants. This has given rise to the development of verification techniques for neural networks, which aim to autonomously verify (i.e., to mathematically prove) that a neural network fulfills a set of correctness properties, thus that it is safe and reliable. However, there is no general definition for safety and reliability of neural networks. In this paper, we start bridging this gap by introducing a benchmark suite for verifying neural networks used to detect anomalies in distillation processes. With this benchmark suite we aim at confronting existing verification methods with complex, 'real-life' properties and thereby foster new advances in the field of neural network verification.

Keywords: Verification of Neural Networks, Benchmark Suite, Distillation Process

1 Introduction

Chemical process engineering is a field of research which investigates material conversion through chemical reactions and how results can be scaled-up from a laboratory level to industrial production processes. This research often includes building and running large pilot plants to experiment with different setups and hyperparameters, to observe the chemical processes under 'real-life' conditions, and to check the safety and reliability of the plant. The success of machine learning in other domains - where it even outperforms human experts - has led to increased use of AI in the field of chemical process engineering as well. One prominent application is anomaly detection, i.e., finding behavior which diverges from the norm. For instance, an anomaly detector could be used as part of the controlling software of a (autonomous) chemical plant or factory to trigger an alarm whenever the system has a malfunction or exhibits unforeseen behavior.

While neural networks show paramount performance in many anomaly detection tasks, they are error-prone in general [10], which is particularly problematic when deployed in safety-critical applications, such as chemical plants. For instance, unreported anomalies may result in imminent hazards to the environment and harm of human life. In contrast, false alarms may lead to substantial financial or scientific loss due to unnecessary downtime of a plant.

The desire of exploiting the capabilities of neural networks even in safety-critical applications has given rise to an increasing demand for methods that can ensure the safety and reliability of neural networks or provide formal guarantees on their behaviour. Inspired by methods from traditional software verification, this led to the development of a variety of verification techniques for neural networks in recent years (see the section on related work for a brief overview). Given a neural network, these methods aim to autonomously verify (i.e., to mathematically prove) that the neural network fulfills a set of correctness properties, ensuring that it is safe and reliable. However, there is no general definition for safety and reliability of neural networks. Hence, most research in this field focuses on a correctness property called adversarial robustness, i.e, the question whether small perturbations of an input can lead to drastic changes in the output. While this is an important property many neural networks lack in general even when they have a really high prediction accuracy, it is insufficient in ensuring the safety and reliability of complex chemical plants.

In this paper, we start bridging this gap by introducing a benchmark suite for neural network verification tools. With this benchmark suite we aim at confronting existing verification methods with complex, 'real-life' properties and neural networks trained on time series data. As most of the existing verification methods were developed for simple specifications and feed-forward networks trained on non-sequential data, this provides crucial insights on how state-of-the-art techniques are suited for verifying such 'real-life' problems and will thereby foster new advances in the field of neural network verification.

Our benchmark suite consists of two parts: First, a set of neural networks trained for anomaly detection in the context of distillation processes. Second, a novel set of correctness properties, also called *specifications*, which we derive

from chemical and physical laws. The tasks of the verification tools will then be to verify that the neural networks do or do not satisfy a given property. This benchmark is publicly available [11].

The outline of our paper is as follows. Next, we will review related work in the field. Afterwards, we will introduce the basic concepts and definitions used throughout the paper in Section 2. We will introduce the distillation processes we consider and the plants we use for our experiments in Section 3. Next, we will give an overview on the neural networks we trained based on our experimental data in Section 4. In Section 5, we will formally introduce our correctness properties before combining them with the neural networks to our benchmark suite in Section 6. We conclude with a discussion and an outlook for future work in Section 7.

Related Works

In recent years, the general topic of verifying neural networks has started to receive increasing attention. This has sparked the development of a plethora of neural network verification tools (e.g., [14, 4, 7, 8]). Many of them however, remained short-lived as they were quickly outperformed by more efficient implementations or more advanced verification techniques. Modern state-of-the-art verification tools, such as α-β-CROWN [20], Marabou [11], and PyRAT [2], often rely on a hybrid approach, efficiently combining ideas from multiple methods.

Initially conceived as a friendly competition among researchers in 2020, the Verification of Neural Network Competition (VNN-COMP) [1] started as an annual event. Its goal is to bring the neural network verification community together to foster new advances in the field and aid the development of more efficient verification techniques. For instance, this has led to the definition of standardized formats for neural networks (ONNX) and specification (VNN-LIB) in verification benchmark suites. Furthermore, it facilitates a fair and objective comparison of state-of-the-art neural network verification tools on cost-equivalent hardware.

To support the VNN-COMP endeavours, many free and publicly available benchmark suites have been proposed to be included in the competition. They include benchmarks for verifying adversarial robustness in different safety critical applications such as traffic signs recognition [17] or condition-based maintenance for aircrafts [12]. Furthermore, they include benchmarks targeted at specific, yet unsupported properties to foster development of new techniques in that direction. These properties include scaling up verification to large networks or robustness of transformer models [19].

Our benchmark suite differs from the existing ones by being, to the best of our knowledge, the first benchmark arising from specifications and data from the field of chemical process engineering. As it contains networks trained on time-series data and specifications containing temporal relations, it poses a new challenge of

[11] https://github.com/simonlutz-tudortmund/Benchmark-Suite-for-Verifying-Neural-Anomaly-Detectors-in-Distillation-Processes

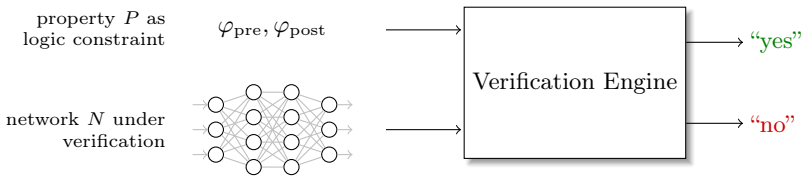

Fig. 1. Verification of Neural Networks

efficiently verifying temporal properties. Furthermore, our specifications derived from chemical and physical laws go beyond the existing, often less complex correctness properties.

2 Preliminaries

In this section, we recapitulate basic notation and introduce definitions used throughout the paper. We start with a brief introduction to the verification of neural networks. Afterwards, we introduce three logical frameworks by defining the syntax and semantics of propositional (Boolean) logic, the First-Order theory of Linear Real Arithmetic, and Linear Temporal Logic (LTL). The latter two will constitute the foundation for the specification language we introduce in Section 5 to formalize our novel set of correctness properties.

2.1 Verification of Neural Networks

While a variety of different verification techniques have been developed in recent years, the general idea is always the same: Given a neural network and a correctness property, a verification engine will compute a formal (i.e., mathematical) proof to show that the network does or does not fulfill the given property (see Figure 1). In order to compute this proof fully automatically, the verification engine requires the correctness property to be provided in a computer-understandable format. Therefore, a common way is to express this property as a set of two constraints in a suitable logical specification language. The first constraint, referred to as *precondition*, typically describes the inputs for which we want to prove correctness. The second constraint, referred to as *postcondition*, then defines the actual property to check. For a more detailed introduction to the verification of neural networks, we refer interested readers to [3].

At this point it is important to stretch that verification of neural networks is orthogonal to the common approach for quality assurance in machine learning. Typically, a neural network is trained on a set of training data and afterwards validated against a large but finite set of unseen test data. The performance of the network is then defined based on the number of test data on which the network predicted the correct outcome. In contrast, verification of neural networks allows us to symbolically argue about an infinite set of inputs, thus providing definitive guarantees on the correctness of the network.

2.2 Propositional Boolean Logic

In the upcoming sections, we introduce three logical frameworks which build the foundation for or can serve as a suitable specification language for formalizing correctness properties for the verification of neural networks. We start with propositional Boolean logic as it is one of the most important concepts in theoretical computer science and the base for both of the subsequent formalisms.

Propositional (Boolean) Logic is a mathematical formalism which allows reasoning about propositions (i.e., statements that are either true or false) and their (logical) relations. In order to formally introduce propositional Boolean logic we use a standard notation of computer science by first introducing its syntax, i.e., describing how to build formulas in the formalism. Afterwards, we introduce the semantics of a formula, i.e., its meaning.

The syntax of a formula in propositional logic is defined by the following inductive definition:
Let Var be a set of propositional variables. Then

- each $x \in Var$ is a formula;
- if Φ and Ψ are formulas, then so are $\neg\Phi$, $\Phi \vee \Psi$, and $\Phi \wedge \Psi$

Furthermore, we can also add the formulas *True*, *False*, $\Phi \rightarrow \Psi$, and $\Phi \leftrightarrow \Psi$ as syntactic sugar. After defining the syntax we will then define the semantics of propositional logic. An *interpretation* $\mathcal{I}$ is a function $\mathcal{I} : Var \mapsto \{0, 1\}$ assigning each variable the value *True* (1) or *False* (0). The semantics of a formula in propositional logic is given by a satisfaction relation $\models$, a binary relation between interpretations and formulas. Intuitively, this relation indicates whether a certain assignment of truth values results in the formula evaluating to true or false. The satisfaction relation $\models$ can also be defined inductively following the syntax: $\mathcal{I} \models x \Leftrightarrow \mathcal{I}(x) = 1$, $\mathcal{I} \models \neg\Phi \Leftrightarrow \mathcal{I} \not\models \Phi$, $\mathcal{I} \models \Phi \vee \Psi \Leftrightarrow \mathcal{I} \models \Phi$ or $\mathcal{I} \models \Psi$, and $\mathcal{I} \models \Phi \wedge \Psi \Leftrightarrow \mathcal{I} \models \Phi$ and $\mathcal{I} \models \Psi$. Following their usual definitions, the semantics can also be extended to the formulas $\Phi \rightarrow \Psi = \neg\Phi \vee \Psi$, and $\Phi \leftrightarrow \Psi = (\Phi \rightarrow \Psi) \wedge (\Psi \rightarrow \Phi)$. We say $\mathcal{I}$ *satisfies* Φ if $\mathcal{I} \models \Phi$ and we call $\mathcal{I}$ a *model* of Φ in that case. In the case that there exists a model $\mathcal{I}$ for a formula Φ, we call Φ *satisfiable*.

While propositional logic allows the reasoning about boolean propositions, this is not sufficient to formalize meaningful safety properties for chemical processes. On the one hand, our data takes on real values, not just propositions *True* and *False*. On the other hand, we consider time series data which also contains temporal relations which cannot be expressed using Boolean operators alone. Therefore, we will consider two extensions of propositional logic which handle real-valued inputs and temporal relations. In Section 5 we will then combine these two to define our specification language for formalizing correctness properties.

2.3 Linear Real Arithmetic

For handling real-valued inputs, we consider the quantifier-free fragment of the First-Order theory of Linear Real Arithmetic (LRA). To a certain extend, LRA

can be seen as an extension of propositional Boolean logic by real-value inputs and basic arithmetic operators. Again, we start by defining the syntax of Linear Real Arithmetic. Before defining a formula in Linear Real Arithmetic, we first introduce the concept of a *term*, a basic building block of a formula. Let Var be a set of real-valued variables, $c \in \mathbb{R}$ be a real-valued constant, and t_1 /t_2 be two terms. Then

$$t ::= x \in Var \mid c \mid c \cdot t_1 \mid t_1 + t_2$$

is a term in Linear Real Arithmetic. Based on this definition, we define a formula in Linear Real Arithmetic as follows: Let t_1 /t_2 be two terms and Φ_1/Φ_2 be two formulas. Then

$$\Phi ::= t_1 = t_2 \mid t_1 < t_2 \mid \neg\Phi_1 \mid \Phi_1 \vee \Phi_2$$

is a formula in LRA.

Furthermore, we allow the arithmetic relations $t_1 \circ t_2$ with $\circ \in \{\neq, \leq, >, \geq\}$ and the boolean combinations *True*, *False*, $\Phi_1 \wedge \Phi_2$, $\Phi_1 \rightarrow \Phi_2$, and $\Phi_1 \leftrightarrow \Phi_2$ as syntactic sugar.

After defining the syntax we will now define the semantics of LRA similar to propositional logic. An *interpretation* $\mathcal{I}$ is a function $\mathcal{I} : Var \rightarrow \mathbb{R}$ assigning each variable a real value. This interpretation can then be lifted to terms as follows: Let $c \in \mathbb{R}$ be a real-valued constant and t be a term. Then $\mathcal{I}(c) = c$, $\mathcal{I}(c \cdot t) = c \cdot \mathcal{I}(t)$, and $\mathcal{I}(t + t') = \mathcal{I}(t) + \mathcal{I}(t')$. The semantics of a formula in Linear Real Arithmetic is given by a satisfaction relation $\models$, a binary relation between interpretations and formulas. This relation can also be defined inductively following the syntax: $\mathcal{I} \models t = t' \Leftrightarrow \mathcal{I}(t) = \mathcal{I}(t')$, $\mathcal{I} \models t < t' \Leftrightarrow \mathcal{I}(t) < \mathcal{I}(t')$, $\mathcal{I} \models \neg\Phi \Leftrightarrow \mathcal{I} \not\models \Phi$, and $\mathcal{I} \models \Phi \vee \Psi \Leftrightarrow \mathcal{I} \models \Phi$ or $\mathcal{I} \models \Psi$. Following their usual definitions, the semantic can also be extended to the formulas $t \circ t'$ with $\circ \in \{\neq, \leq, >, \geq\}$, $\Phi_1 \wedge \Phi_2$, $\Phi_1 \rightarrow \Phi_2$, and $\Phi_1 \leftrightarrow \Phi_2$. We say $\mathcal{I}$ *satisfies* Φ if $\mathcal{I} \models \Phi$ and we call $\mathcal{I}$ a *model* of Φ in that case. In the case that there exists a model $\mathcal{I}$ for a formula Φ, we call Φ *satisfiable*.

2.4 Linear Temporal Logic

To handle temporal relations in our correctness properties we rely on *Linear Temporal Logic* (LTL) [16] which extends propositional logic by a set of temporal operators. Similar to propositional logic the syntax can be defined by an inductive definition: Let Var be a finite, nonempty set of propositional variables. Then

- each variable $x \in Var$ is an LTL formula;
- if Φ and Ψ are formulas, so are $\neg\Phi$, $\Phi \vee \Psi$, XΦ (“next”), and ΦUΨ (“until”)

As in the case of propositional logic we can add the formulas *True*, *False*, $\Phi \wedge \Psi$, $\Phi \rightarrow \Psi$, and $\Phi \leftrightarrow \Psi$ as syntactic sugar. Furthermore, we can also add the temporal operators FΦ (“finally”) and GΦ (“globally”). In contrast to propositional logic, formulas in linear temporal logic are interpreted over (infinite) sequences $w \in (2^{Var})^{\omega}$ of sets over the variables. Intuitively, each position of the

sequence describes the set of variables which are $True$ at this point in time. The semantics of LTL is also defined by the means of an *interpretation* $\mathcal{I}$. In the context of LTL, an interpretation is a function $\mathcal{I} : (\Phi, w) \mapsto \{0, 1\}$ mapping pairs of LTL formulas and infinite sequences to Boolean values. With Φ and Ψ being formulas in Linear Temporal Logic, this function is inductively defined by: $\mathcal{I}(x, w) = 1 \Leftrightarrow x \in w_0$ for $x \in Var$, $\mathcal{I}(\neg\Phi, w) = 1 - \mathcal{I}(\Phi, w)$, $\mathcal{I}(\Phi \vee \Psi, w) = \mathcal{I}(\Phi, w)$ or $\mathcal{I}(\Psi, w)\}$, $\mathcal{I}(\mathrm{X}\Phi, w) = \mathcal{I}(\Phi, w[1, \infty))$, and $\mathcal{I}(\Phi \mathrm{U} \Psi, w) = \max_{i \geq 0}\{\min\{\mathcal{I}(\Psi, w[i, \infty)), min_{0 \leq j < i}\{\mathcal{I}(\Phi, w[j, \infty))\}\}\}$, where w_0 describes the first position of a sequence w and $w[i, \infty)$ describes the sequence w but starting at position i. Following the syntax this definition can be also extended to the formulas *True*, *False*, $\Phi \wedge \Psi$, $\Phi \rightarrow \Psi$, $\mathrm{F}\Phi := True \mathrm{U} \Phi$, and $\mathrm{G}\Phi := \neg\mathrm{F}(\neg\Phi)$. Intuitively, the temporal formula $\mathrm{X}\Phi$ encodes that the formula Φ needs to hold at the next position in time. Furthermore, the formula $\Phi \mathrm{U} \Psi$ encodes that at some point in the future the formula Ψ needs to hold and on every point until then the formula ϕ holds. Moreover, the formulas $\mathrm{F}\Phi$ and $\mathrm{G}\Phi$ indicate that Φ needs to hold at some or all points in time, respectively. We say w *satisfies* Φ if $\mathcal{I}(\Phi, w) = 1$ and call $\mathcal{I}(\Phi, w)$ the *interpretation of* Φ *under* w.

3 Data Generation and Experimental Setup

In this work we focus on verifying anomaly detectors for distillation processes. Simplified, distillation is a chemical processes which uses a so-called distillation column to separate a liquid mixture into multiple, possibly unknown chemical components. In the distillation column, the liquid mixture is heated up in the distillation still, a vessel connected to a heat source. When the temperature exceeds the boiling temperature of the lowest boiling component in the mixture it starts to vaporize. The resulting vapor rises up through the rectifying column, containing a sequence of packings or plates, until it is condensated in the condenser at the top of the column. The condensate is cooled and split into two separate streams; a part of the condensate is withdrawn as distillate while the other, often larger fraction of the condensate called reflux is returned to the column top. There, the countercurrent of downflowing liquid and uprising vapor is constantly in contact, enabling a mass transport between the phases improving the separation of the components in the mixture [5]. By adjusting, e.g., pressure, temperature and ratio of withdrawn distillate, the purity of the distillate or residue can be influenced in the process.

In chemical engineering one typically distinguishes between continuous and discontinuous processes. In continuous processes the plant usually operates nonstop, meaning 24 hours per day, seven days per week, and is only interrupted for maintenance. These plants are widely used in the chemical industry, e.g., petrochemical industry, as they are flexible large-scale, highly efficient, and cost-effective solutions to consistently produce large product volumes. In contrast, discontinuous, or batch, processes run only for a certain time frame. Such processes are in general used for small-scale production units with high product-quality requirements, e.g. in pharmaceutical or bio-chemical industries, as well

as in scale-up studies. Batch processes are widely adaptable to many process requirements and are not fixed to a single operating point. Their dynamic, thus flexible, nature allows for swift reaction to unexpected process behavior.

In our work, we consider both a discontinuous, batch distillation process and a continuous distillation process. In the following, we will describe the plants we use, the setup for collecting data, and the experiments we conducted[12].

3.1 Setup of the Batch Distillation Plant

The experimental data was collected using a multi-stage batch distillation unit of the type LM 2/S provided by Iludest® (*ILUDEST Destillationsanlagen GmbH*, Waldbüttelbrunn, Germany). The batch distillation plant is a combination of a heated round-bottom flask with a volume of $V = 2$ L with a put-on glass rectifying column. The rectifying column consists of 3 sections with a length of 50 cm each and an inner diameter of 50 mm. Each column section contains structured laboratory packing of the type Sulzer DX30 (*Sulzer*, Winterthur, Switzerland). The sections as well as the upper hemispherical part of the bottom flask, are equipped with a glass fibre-insulated heating jacket to supply additional heating for the minimization of heat losses and the establishment of a quasi-adiabatic process environment. An image of our batch distillation plant is displayed in Figure 2.

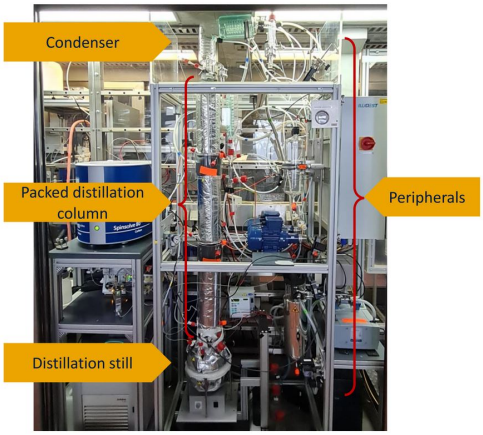

Fig. 2. Our discontinuous, batch distillation plant at the University of Kaiserslautern-Landau used to generate the data

The distillation plant is rigged with a plethora of sensors. Temperatures are measured at the bottom of each column section, as well as in the reboiler and at the top of the column; moreover, the temperatures of the heating jackets are

[12] We will also make this experimental data publicly available.

monitored at all times. At the top of the column, a condenser is installed to condensate the uprising vapor. The reflux ratio, another important parameter controlling the operating point of the plant [18], is set using two pumps situated downstream of the condenser, whereas one pump controls the distillate flux and the other the reflux into the plant, which are also measured. The batch distillation plant operates under vacuum conditions; the internal pressure is thereby measured at the top of the column. Moreover the differential pressure between condenser and bottom still is measured. In addition, the system is connected to a benchtop nuclear magnetic resonance (NMR) spectroscope of the type Spinsolve 80 ULTRA Carbon (*Magritek*, Aachen, Germany) to measure distillate and residual compositions. In addition to the NMR device, samples are drawn from bottom still and distillate receiver in discrete time intervals to allow composition measurements with gas chromatography.

Experiments For distillation experiments, a feed mixture of 50 mol-% 2-propanol, 25 mol-% 1-butanol and 25 mol-% water is supplied in the bottom still of the batch distillation plant. The feed mixture is heated under set vacuum conditions, the uprising vapor is condensed and a part of that condensate is removed as distillate, while the remains flow down though the column into the bottom still. This process is maintained until 500 ml of distillate were drawn in each experiment. During operation of the batch-distillation plant, all sensors and actors mentioned above are monitored and recorded by the process control software. As anomalies are rare in the normal operation of an experiment and could cause real danger if undetected, we introduced them in a controlled and clearly-defined manner. We started by reviewing literature on the most common process anomalies from real-life processes to support the optimal selection of recreated anomalies [13]. Based on that review, a location of attack was defined in the process - either a certain control loop, sensor or component of the plant. Then, for a short window of time (about 10 minutes) during an experiment, we introduced a fault into the process, for instance, a leakage, a sealing failure, or a control setting deviating from normal behavior.

3.2 Setup of the Continuous Distillation Plant

Our continuous distillation plant has a feed capacity of 5 tons per annum (t/a) and comprises two distillation columns – a steel column (DN70 specification), a glass column (DN50 specification) and a decanter vessel. The plant is manufactured by Iludest® (*ILUDEST Destillationsanlagen GmbH*, Waldbüttelbrunn, Germany) and both the columns have packing manufactured by Sulzer (*Sulzer*, Winterthur, Switzerland). An image of our continuous distillation plant is displayed in Figure 3. This intricate system has 49 sensors, including temperature, pressure, flow, level sensors, and mass scales. They are all connected to LabVIEW, our primary tool for data collection and controlling of the plant. In addition to these sensor readings, we conduct offline analyses such as gas chromatography (GC) and nuclear magnetic resonance (NMR) to determine the composition of components at different stages of the column.

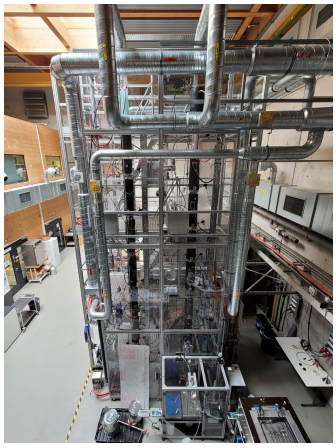

Fig. 3. Our continuous distillation plant at the Technical University of Munich used to generate the data

Experiments In our experiments, chemical systems of increasing complexities were implemented in the plant. We started with a single-component water system and one distillation column. For this system, continuous experimental data was generated for 30 days, at operating pressures between 1 bar to 1.4 bar. Afterwards, a hetero-azeotropic binary system of n-butanol and water is used to run the plant, involving two columns and a decanter. This system involves an azeotrope formation and two liquid phases separated using the decanter. We experimented with different feed concentrations, ranging from 90 to 98 % water and 10 to 2% n-butanol, respectively. This feed is continuously fed into the first column, where the bottom product is expected to be pure water and the condensate at the top from the first column is an azeotrope which is collected in the decanter. This has two liquid phases, from which the lighter phase or the n-butanol rich phase is fed into the second column at the top stage. The heavier or water rich phase is fed back into the first column as reflux. The bottom product of the second column is expected to be pure n-butanol and the top product is an azeotrope which is also fed into the decanter. Both the columns are operated at atmospheric pressure and data is generated every 30 seconds with and without anomalies. Similar to the experiments with the batch distillation plant, we manually introduced anomalies inspired by anomalies from real-life processes. These anomalies include faulty sensor readings, pipe clogging, catalyst deactivation, feed irregularities, and level discrepancies. For instance, we reproduced pipe clogging by manipulating some of the hand valves, thereby restricting the flow.

4 Neural Networks for Anomaly Detection

In this section, we introduce the neural networks we trained on our experimental data to detect anomalies in continuous and discontinuous distillation processes. At this point, it is important to point out that these networks are not trained to achieve state-of-the-art performance but to serve as part of a benchmark suite for neural network verification. This implies that the networks are much smaller and trained with a much simpler architecture than widely used anomaly detectors such as long short term memory networks (LSTMs) [15] or generative adversarial networks (GANs) [9]. This restriction is due to the (yet) limited support of neural network verification tools which often support only fully connected neural networks with piecewise linear activation functions. However, this benchmark can easily be extended to adapt to future advances in the field and the development of new verification techniques.

For both continuous and discontinuous distillation processes we trained six different networks varying in size and the amount of data they where trained on. In general all networks follow the same simple architecture: They expect a multimodal time series of fixed length as input and predict for the last point of the series whether it is an anomaly or not, indicated by one of two output neurons being active. In the hidden layers, the networks consist of a fully connected architecture with ReLU activation functions.

In order to compile a benchmark suite with varying difficulty we trained our networks to detect anomalies in different amounts of sensor readings. Therefore, we trained our most simple anomaly detectors just on a single measurement, the absolute pressure at the top of the packings column. For the next set of anomaly detectors, we increase the amount of sensor readings to four, for instance including temperature readings in the column. Then, the last anomaly detectors are trained on the full set of sensor readings including all 34 (batch) and 49 (continuous) sensors, respectively. In addition to varying amounts of sensor readings, we also trained anomaly detectors of two different sizes. The first set of anomaly detectors was trained with just two hidden layers consisting of 512 neurons each. In the second set, we increased both the number of hidden layers (to four) and the number of neurons per layer (to 1024).

5 Defining Specifications for Distillation Processes

In this section, we introduce the novel set of correctness properties we included in our verification benchmark. We start with an overview of the origin of these specifications, provide some examples and briefly discuss similarities and differences between properties for batch vs. continuous distillation. Then, we formally introduce the specification language we use for formalizing the correctness properties and provide some example specifications. In the end, we conclude with a discussion on the limitations of the VNN-LIB format, currently used for such verification benchmarks, and how to overcome some of them in our specific application.

5.1 Specifications for Distillation Processes

For many decades, researchers in the field of chemical process engineering have studied distillation processes and developed a detailed understanding of the chemical and physical laws influencing these processes. This allows them to formally and accurately describe the expected, thus normal, behaviour during operation of a chemical plant. We exploit this rich domain knowledge to develop correctness properties for anomaly detectors. In a nutshell, our specifications encode that whenever a chemical or physical law is violated the anomaly detector should predict an anomaly. This allows the specifications to function as a sanity check for the neural anomaly detectors, ensuring that the models are at least conform with the physical and chemical laws.

Note that the output of the anomaly detector is always the same for all our specifications (it predicts an anomaly, if a chemical or physical law is violated). Therefore, we will, for now, focus on just specifying the chemical and physical laws. We will later see how to also express the behaviour of the neural network.

We can categorize our specifications into four sets: The first set consists of specifications which define a certain bound on the measurements, for instance, "The pressure must always be greater than zero", "The measured amount of a substance (i.e., its concentration) must always be greater than or equal to zero", or "The temperature in the plant must not exceed a certain threshold T" (this threshold may vary from plant to plant). The second set of specifications defines relations between two measurements, either of different sensors or at different points of time. These include properties like "The temperature has to decrease from the bottom of the packing column to the top", "The composition of the low boiler must be higher at the top of the column compared to the bottom" (for zeotropic mixtures), or the balance equation, i.e., "The sum of the concentrations of all substances in the mixture cannot exceed one". While the above properties hold for all distillation processes, due to their different nature there are also properties which only hold for continuous or discontinuous processes. Our third and fourth set of constraints therefore contain the specifications which exclusively hold for continuous and discontinuous processes, respectively. They include, for instance, "In steady state, all of the process variables (temperature, pressure, concentrations, and flow rates) remain constant over time" (exclusive for continuous distillation) or "The temperature in the distillation still increases over time" (exclusive for batch distillation). To not clutter this section too much we omit presenting the full list of specifications at this point and refer interested readers to the appendix.

5.2 A Specification Language for Neural Networks Trained on Time Series Data

In order to verify that a neural network fulfills the specifications defined above, we need to transform them from natural language to a representation an autonomous verifier, i.e. a computer program, can interpret and process. A closer inspection of the specifications shows that we need to express constraints and

bounds on real-valued variables, for instance, the value of a temperature sensor. Furthermore, we need to be able to relate measurements of different points in time and formalize that a property has to always hold, thus temporal behavior. To capture both these properties, we rely on a combination of Linear Real Arithmetic and Linear Temporal Logic when formalizing the specifications.

As for the logical frameworks introduced in Section 2, we first define the syntax of our specification language. We start by defining the concept of a term, a building block for our formulas. Let Var be a set of real-valued variables, $c \in \mathbb{R}$ be a real-valued constant, and t_1 /t_2 be two terms. Then

$$t ::= x \in Var \mid c \mid c \cdot t_1 \mid t_1 + t_2 \mid \bigcirc\!\!\!\!\sim x$$

is a term in our specification language. Intuitively, this just extends the notion of terms in Linear Real Arithmetic by the new function $\bigcirc\!\!\!\!\sim x$. Based on this definition, we define a formula in our specification language as follows: Let t_1 /t_2 be terms and φ_1/φ_2 be formulas. Then

$$\varphi ::= t_1 = t_2 \mid t_1 < t_2 \mid \neg\varphi_1 \mid \varphi_1 \vee \varphi_2 \mid \mathrm{X}\varphi \mid \varphi \mathrm{U} \psi$$

is a formula in our specification language. Note that we denote formulas in our specification language by small Greek letters to differentiate them from formulas in another logical formalism. After defining the syntax, we will define the semantics of our specification language next. Similar to Linear Temporal Logic, our specifications are interpreted over sequences, but over sets of real values (instead of sets of propositions). Intuitively, one of these sets represents the sensor values at a specific point in time. The semantics of our specification language is defined by means of an interpretation $\mathcal{I}$. In the context of our specification language, an interpretation is a function $\mathcal{I} : (\varphi, w) \to \{0, 1\}$ mapping pairs of formulas and sequences to Boolean values. This function is inductively defined following the usual definitions for boolean connectives $\neg$ and $\vee$, temporal operators X and U, and arithmetic functions and relations $+, \cdot, =$, and $<$. The function $\bigcirc\!\!\!\!\sim x$ will refer to the value of a variable at the following position of a sequence, i.e., $I(\bigcirc\!\!\!\!\sim \mathrm{x}, w_1, \ldots, w_n) = w_2$.

After introducing the specification language we will now display the formal representation of some of our specifications. Recall that we place numerous sensors along the distillation column measuring, for instance, temperature, pressure, or concentration of a component s. In the following, we will denote them by t_i, p_i, and c_i^s, respectively. The index will indicate the position of the sensor within the column where the sensor x_0, for $x \in \{t, p, c\}$, is placed at the bottom and the sensor x_{n_x} is placed at the top of the packings column. For simplicity, the upcoming examples assume a plant with a single packings column. We can extend them to multiple columns in a straightforward way by using superscripts x^1 and x^2 to differentiate between senors in different columns. To not clutter this section too much we will only display a two example specifications and refer

interested readers to the appendix for the full list.

informal: The pressure must always be greater than zero.

formal: $\mathrm{G}(\bigwedge_{i=0}^{n_p} p_i > 0)$

informal: The temperature has to decrease from the bottom of the packings column to the top

formal: $\mathrm{G}(\bigwedge_{i=0}^{n_t-1} t_i > t_{i+1})$

After describing how we can formulate chemical and physical laws in our specification language, we will now explain how we can also formalize full specifications. Recall that we want to express properties of the form "whenever a chemical or physical law is violated the anomaly detector predicts an anomaly". Let φ be a formula defining a specific chemical or physical law. Then we can formalize the above correctness property as

$$\neg\varphi \rightarrow N(\overrightarrow{\mathbf{x}}) = 1$$

where $N(\overrightarrow{\mathbf{x}}) = 1$ denotes that the neural network outputs "anomaly" on the given input $\overrightarrow{\mathbf{x}}$.

5.3 Specifications for our Benchmark Suite

In the previous section, we introduced our specification language for formalizing chemical and physical laws in continuous and discontinuous distillation processes. While this specification language allows us to express a large and meaningful set of correctness properties, most of the neural network verification tools today are not yet designed to verify temporal properties. Instead, they rely on specifications provided in the VNN-LIB format [6], an initial attempt for standardizing the description of neural networks and specifications for verification. So far, this format only supports specifications defined in Linear Real Arithmetic. While this is a issue when considering correctness properties which include temporal behaviour, we can overcome this restricting in our specific application. Recall that our neural networks require an input of a fixed input length (typically the values of the last k measurements). We can exploit this fact to transform our temporal constraints into specifications only expressed in Linear Real Arithmetic. Towards this goal, we introduce a real-valued variable x_i^j for every sensor x_i and every time point in our fixed input length. Intuitively, this variable x_i^j will then store the measurement of sensor x_i at time j. This allows us to model temporal behaviour by explicitly referencing the corresponding variables. For instance, we transform the specification stating that "The pressure must always be greater than zero" into the following constraint in LRA:

$$\mathrm{G}(\bigwedge_{i=0}^{n_p} p_i > 0) \quad \Rightarrow \quad \bigwedge_{j=1}^{k} \bigwedge_{i=0}^{n_p} p_i^j > 0$$

6 Compiling the Benchmark Suite

In this section, we introduce the benchmark suite for neural network verification tools we compiled by combining the neural networks we trained on our experimental data and the specifications we elicited from chemical and physical laws. Every instance in the benchmark suite will consist of a neural network and a correctness property. The task of the verification tool is to verify whether the given network does or does not fulfill the respective property. To make this benchmark as accessible as possible, we will use the standardized formats for verification benchmarks and represent the neural networks and the specifications in the ONNX and VNN-LIB file format, respectively. Furthermore, this benchmark suite is publicly available and we will propose to include it in further iterations of the Verification of Neural Network Competition.

In order to provide a more detailed insights into the state of the art of verifying temporal properties and to better monitor future advances, we propose a benchmark suite including instances of varying difficulty. On the one hand, this is achieved by including neural networks of different size and input (see Section 4). On the other hand, we will also provide a simplified version of the specifications discussed in Section 5. To provide an example of such a simplified specification, consider the physical law stating that "The pressure must always be greater zero". Then, this law is violated whenever an input sequence contains a nonpositive pressure measurement (e.g., due to a sensor malfunction). This can easily be expressed in our specification language using the constraint $\bigvee_{i=0}^{n_p} p_i \leq 0$. However, this implies that the nonpositive value can occur at every position in the sequence, resulting in a very large search space. One possible way of simplifying this specification could be to specify the exact position in the sequence where the negative occurs (e.g., $p_4 \leq 0$) or even simpler to constrain the input to have a specific negative value at a certain position ($p_4 = -0.53$). Note that the search space is still infinite in the first case but easier to traverse by the verification tool. The benchmark suite will then contain every suitable combination of network and (simplified) specification together with the expected outcome of the verifier.

7 Conclusion and Future Work

In this work, we introduced a benchmark suite for verifying neural anomaly detectors in distillation processes. This benchmark suite consists of a set of neural networks trained to detect anomalies in time-series data collected from continuous and discontinuous distillation processes. Furthermore, it consists of a novel set of correctness properties derived form chemical and physical laws which need to be fulfilled during the normal, fault-free operation of a distillation plant. With this benchmark suite we aim at confronting existing verification methods with complex, 'real-life' properties and will thereby foster new advances in the field of neural network verification.

As part of future work, we plan to extent this benchmark suite in three directions. First, we plan to add additional neural networks to the benchmark suite including more sophisticated techniques and state-of-the-art architectures. As most verification tools only support a restricted set of neural network architectures, these methods could give a direction for future advances in the field and the development of novel verification techniques. Second, we will investigate more expressive specification languages. While our specification language is able to formalize a wide variety of correctness properties, it is restricted to specifications which can be expressed as linear constraints. As some chemical and physical laws in a distillation process are highly non-linear in nature, this may not be sufficient to formalize them all, thus requiring a more expressive specification language. In this regard, we will need to carefully investigate the algorithmic properties of such a more expressive specification language as there is a trade-off between the expressiveness of a logic and its algorithmic properties in general. This could render the verification task to become infeasible. Last, we plan to continue the process of eliciting and formalizing correctness properties for other chemical processes besides continuous and discontinuous distillation. This will foster the development of novel verification techniques and ultimately ensure the safety and reliability of neural networks used in chemical process engineering.

Acknowledgements This work has been financially supported by Deutsche Forschungsgemeinschaft, DFG Project number 459419731, and the Research Center Trustworthy Data Science and Security (https://rc-trust.ai), one of the Research Alliance centers within the UA Ruhr (https://uaruhr.de).

References

1. 5th international verification of neural networks competition (vnn-comp'24) website, https://sites.google.com/view/vnn2024 [Accessed: June 22nd, 2024]
2. Pyrat analyzer website, https://pyrat-analyzer.com/ [Accessed: June 22nd, 2024]
3. Albarghouthi, A.: Introduction to neural network verification (2021)
4. Bak, S.: nnenum: Verification of relu neural networks with optimized abstraction refinement. In: NASA Formal Methods Symposium. pp. 19–36. Springer (2021)
5. Biegler, L.T., Grossmann, I.E., Westerberg, A.W.: Systematic methods for chemical process design (12 1997), https://www.osti.gov/biblio/293030
6. Demarchi, S., Guidotti, D., Pulina, L., Tacchella, A.: Supporting standardization of neural networks verification with vnn-lib and coconet. In: Proceedings of the 6th Workshop on Formal Methods for ML-Enabled Autonomous Systems. vol. 16, pp. 47–58 (2023)
7. Duong, H., Li, L., Nguyen, T., Dwyer, M.: A dpll (t) framework for verifying deep neural networks. arXiv preprint arXiv:2307.10266 (2023)
8. Ferlez, J., Khedr, H., Shoukry, Y.: Fast batllnn: fast box analysis of two-level lattice neural networks. In: Proceedings of the 25th ACM International Conference on Hybrid Systems: Computation and Control. pp. 1–11 (2022)

9. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks **63**(11), 139–144 (oct 2020). https://doi.org/10.1145/3422622, https://doi.org/10.1145/3422622
10. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples (2015)
11. Katz, G., Huang, D.A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljić, A., et al.: The marabou framework for verification and analysis of deep neural networks. In: Computer Aided Verification: 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I 31. pp. 443–452. Springer (2019)
12. Kirov, D., Rollini, S.F.: Benchmark: remaining useful life predictor for aircraft equipment. In: International Conference on Bridging the Gap between AI and Reality. pp. 299–304. Springer (2023)
13. Kister, H.Z.: What caused tower malfunctions in the last 50 years? Chemical Engineering Research and Design **81**(1), 5–26 (2003)
14. Lopez, D.M., Choi, S.W., Tran, H.D., Johnson, T.T.: Nnv 2.0: the neural network verification tool. In: International Conference on Computer Aided Verification. pp. 397–412. Springer (2023)
15. Malhotra, P., Vig, L., Shroff, G., Agarwal, P.: Long short term memory networks for anomaly detection in time series (04 2015)
16. Pnueli, A.: The temporal logic of programs. In: 18th Annual Symposium on Foundations of Computer Science (sfcs 1977). pp. 46–57 (1977). https://doi.org/10.1109/SFCS.1977.32
17. Postovan, A., Eraşcu, M.: Architecturing binarized neural networks for traffic sign recognition. arXiv preprint arXiv:2303.15005 (2023)
18. Seader, J.D., Henley, E.J., Roper, D.K.: Separation process principles: Chemical and biochemical operations. Wiley, Hoboken NJ, 3rd ed. edn. (2011)
19. Shi, Z., Jin, Q., Kolter, J.Z., Jana, S., Hsieh, C.J., Zhang, H.: Formal verification for neural networks with general nonlinearities via branch-and-bound. 2nd Workshop on Formal Verification of Machine Learning (WFVML 2023) (2023)
20. Zhang, H., Weng, T.W., Chen, P.Y., Hsieh, C.J., Daniel, L.: Efficient neural network robustness certification with general activation functions. Advances in neural information processing systems **31** (2018)

Appendix

The following displays the full list of specifications and their formal representation. We will denote the sensor readings for temperature, pressure, concentration of a substance s, (filling) level, liquid flow rate, and vapor flow rate by variables t_i, p_i, c_i^s, ℓ_i, f_i^ℓ, and f_i^v, respectively.

1. Specifications defining bounds (on measurements):

- The pressure must always be greater than zero.

$$\mathrm{G}(\bigwedge_{i=0}^{n_p} p_i > 0)$$

- The measured amount of substance (i.e., its concentration c^s) must always be greater or equal to zero

$$\mathrm{G}(\bigwedge_{s \in M} \bigwedge_{i=0}^{n_c} c_i^s \geq 0)$$

- The cooling Q_C has to be negative

$$\mathrm{G}(Q_C < 0)$$

- The temperature in the plant must not exceed a certain threshold T

$$\mathrm{G}(\bigwedge_{i=0}^{n_t} t_i < T)$$

- The pressure in the plant must not exceed a certain threshold P

$$\mathrm{G}(\bigwedge_{i=0}^{n_p} p_i < P)$$

2. Specifications defining Relations (between two measurement):

- The pressure at the bottom of the packings column p_0 is greater than the pressure at the top p_{n_p}

$$\mathrm{G}(p_0 > p_{n_p})$$

  Remark: If the distillation column only consists of a single column the pressures may also be equal and the above specification must be adjusted

– The temperature at the bottom of the packings column t_0 is greater than the temperature at the top t_{n_t}

$$\mathrm{G}(t_0 > t_{n_t})$$

Remark: If the distillation column only consists of a single column the temperatures may also be equal and the above specification must be adjusted

– The temperature has to decrease from the bottom of the packings column to the top

$$\mathrm{G}(\bigwedge_{i=0}^{n_t-1} t_i > t_{i+1})$$

Remark: If the distillation column only consists of a single column or to sensors are placed in the same packing the temperature may also be equal and the above constraint must be adjusted

– The concentration of the low boiler c^{LS} (i.e., the substance with the lowest boiling point) has to be higher in the condenser c_0^{LS} than at the bottom of the packings column $c_{n_c}^{LS}$ (for zeotropic mixtures).

$$\mathrm{G}[(c_0^{LS} < c_{n_c}^{LS})]$$

The amount of condensate $m_{Brüden}$ is at least the amount of reflux m_{reflux}

$$\mathrm{G}(m_{Brüden} \geq m_{reflux})$$

– For each stage, the balance equation has to hold, i.e., the sum of the concentrations of all substances s in the mixture M can not exceed 1

$$\mathrm{G}(\bigwedge_{i=0}^{n_c} \sum_{s \in M} c_i^s \leq 1)$$

3. Specifications exclusive for continuous distillation:

– For each component s, the total mass entering each column i ($m_{feed_i}^s$) must be greater than or equal to the total mass leaving as distillate ($m_{distillate_i}^s$) and bottoms ($m_{bottoms_i}^s$).

$$\mathrm{G}(m_{feed_i}^s \geq m_{distillate_i}^s + m_{bottoms_i}^s)$$

– For each component, the total mass entering the overall system (m_{feed}^s) must be greater than or equal to the total mass leaving the system as product ($m_{product}^s$).

$$\mathrm{G}(m_{feed}^s \geq m_{product}^s)$$

– The level in each of the column ℓ^i should at least be a certain minimum volume L^{min} and and not exceed a certain maximum volume L^{max}

$$\mathrm{G}(L^{min} \leq \ell^i \wedge \ell^i \leq L^{max})$$

– In steady state, all of the process variables (temperature, pressure, concentrations, and flow rates) remain constant over time. For $x \in \{t, p, c^s, f^\ell, f^v\}$:

$$\mathrm{G}(\bigwedge_{i=0}^{n_x} x_i = \circledcirc x_i)$$

Note that this condition may be too strict for practice, as small disturbances may occur even in steady state. Therefore, we can relax the condition by allowing subsequent measurements to change by a small predefined value ε.

$$\mathrm{G}(\bigwedge_{i=0}^{n_x} x_i - \varepsilon \leq \circledcirc x_i \wedge \circledcirc x_i \leq x_i + \varepsilon)$$

– For each stage i, the component balance must be maintained. Let L_i, V_i, and F_i denote the liquid, vapor and feed flow rate of stage i, respectively. Furthermore, let x_i, y_i, and z_i denote the liquid, vapor and feed composition at stage i.

$$\mathrm{G}(L_{i+1}x_{i+1} + V_{i-1}y_{i-1} + F_i z_i = L_i x_i + V_i y_i)$$

– The vapour and liquid compositions must be in equilibrium in each stage i as represented by the VLE data. Let x_i, and y_i denote the liquid and vapor composition at stage i.

$$\mathrm{G}(y_i = \{_{vle,i}(p_i, t_i, x_i) = a \cdot x_i)$$

Note that the function $\{_{vle,i}(p_i, t_i, x_i)$ is non-linear in general. However, during operation in remains constant and can therefore be precomputed in advance. This allows us to simplify the constraint using a suitable constant a.

– In each stage i of the column, the heat balance (enthalpy balance) must be satisfied. Let h_i^L, h_i^V, and h_i^F denote the liquid, vapor and feed enthalpy at stage i. Furthermore, let Q_i be the cooling at stage i.

$$\mathrm{G}(L_{i+1}h_{i+1}^L + V_{i-1}h_{i-1}^V + F_i h_i^F = L_i h_i^L + V_i h_i^V + Q_i$$

4. Specifications exclusive for discontinuous, batch distillation:

– The level ℓ_0 (i.e., the amount of mixture) in the distillation still has to be positive

$$\mathrm{G}(\ell_0 \geq 0)$$

– For known mixtures the pressure has to be within a given interval $[p_i^{min}, p_i^{max}]$

$$\mathrm{G}(\bigwedge_{i=1}^{n} p_i^{min} \leq p_i \wedge p_i \leq p_i^{max})$$

– The amount of product $m_{product}$ can not exceed the amount of input m_{input} as parts of the mixture remain in the plant

$$\mathrm{G}(m_{input} > m_{product})$$

– The mass balance has to be fulfilled in the end, i.e., the sum of the product $m_{product}$ and the remainder in the distillation still $m_{distilationstill}$ equals the amount of input m_{input}. Note that this condition is to strict for practice, as small amounts of liquid remain in the distillation column. Therefore, we relax the condition to greater or equal instead of strictly equal.

$$\mathrm{G}(\mathrm{X}false \rightarrow (m_{input} \geq m_{product} + m_{distillationstill}))$$

– Continuity over time for pressure, temperature and concentration. Note that we verify continuity by checking whether the difference between the measurements is smaller than some predefined threshold ϵ. With $x \in \{p, t, c\}$

$$\mathrm{G}(\bigwedge_{i=0}^{n_x} x_i - \epsilon_x \leq \circledcirc x_i \wedge \circledcirc x_i \leq x_i + \epsilon_x)$$

– Continuity over the packing column for pressure and temperature. With $x \in \{p, t\}$

$$\mathrm{G}(\bigwedge_{i=0}^{n-1} x_i - \epsilon_x \leq x_{i+1} \wedge x_{x+1} \leq x_i + \epsilon_x)$$

5. More specifications which do not have a formal representation (yet):

– Energy balance over the whole distillation (1. law of thermodynamics):

$$\text{sum of heating} = \text{sum of cooling} + \varepsilon$$

– The chemical process has to follow certain constraints, e.g., distillation lines (dependent of the mixture): In batch distillation processes, there is a non-linear function for each component which describes its concentration a each time point during the distillation
– The amount of condensate ($m_{Brüden}$) has to correspond to the amount of heating in the distillation still: In batch distillation processes, there is a non-linear function also depending on concentration and pressure.
– The fluid dynamic restrictions of the packings must be complied with, e.g. gas & liquid load or maximal capacity: These restrictions are described by non-linear functions
– If $m_{Ruecklauf} = 0$ or after really long time, the measurements in the distillation still and at the top of the packing column of batch distillation plants are connected through phase equilibrium conditions: These conditions are described by non-linear functions