

Frontend-Entwicklung eines Reduktions-Trainers für Studierende

Bachelorarbeit
zur Erlangung des akademischen Grades Bachelor of Science

von

Clemens Weiße

Matrikelnummer: 33235923

Vorgelegt im: Fachgebiet Theoretische Informatik/
Formale Methoden

Gutachter: Prof. Dr. Martin Lange
Prof. Dr. Albert Zündorf

Betreuer: Dr. Norbert Hundeshagen

Eingereicht am: 20. Januar 2023

Eigenständigkeitserklärung

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken (dazu zählen auch Internetquellen) entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht.

Kassel, den 20. Januar 2023

Inhaltsverzeichnis

1	Einleitung	3
2	Vorwissen	4
2.1	Berechenbarkeit	4
2.2	Entscheidbarkeit und Entscheidungsprobleme	4
2.3	Reduktion	6
3	High-Level Beschreibung des Reduktions-Trainers	8
3.1	Aufbau des Trainers	8
3.2	Studierenden-Frontend	9
3.3	Dozenten-Frontend	12
3.4	Datenbank Backend	14
4	Umsetzung	15
4.1	Benutzeroberfläche für Studierende	15
4.2	Benutzeroberfläche der Dozenten	33
4.3	Backend und Datenbank	54
5	Fazit	58

1 Einleitung

Zu lernen wie eine Reduktion funktioniert und diese auch anzuwenden, ist für viele Studierende der Informatik Teil des Grundstudiums. Beweise in der Theoretischen Informatik zu führen, fällt vielen dieser Studierenden schwer ([2], [3]). Dabei sind Reduktionsbeweise keine Ausnahme. Während meines Studiums und auch während ich als studentische Hilfskraft im Fachgebiet Theoretische Informatik / Formale Methoden gearbeitet habe, bin ich immer wieder damit in Kontakt gekommen, wie schwer Studierenden Reduktionsbeweise fallen. Da das Fachgebiet Interesse an weiteren Programmen hat, welche die Studierenden beim Lernen unterstützen und ich selbst auch schon beim Lernen von diesen Programmen profitiert habe, wurde ich zum Vorhaben, ein Frontend für einen Reduktions-Trainer zu entwickeln, ermutigt.

Diese Ausarbeitung erläutert einen Teil der Entwicklung eines Reduktions-Trainers. Der entwickelte Reduktions-Trainer besteht aus zwei Ausarbeitungen, die andere ist von Kathrin Lehmann [7]. Während es in der Arbeit von Kathrin Lehmann darum geht, wie Reduktionscode automatisch ausgewertet werden kann, befasst sich diese Arbeit damit wie der Reduktions-Trainer Studierenden hilft, das Umsetzen von Reduktionen besser zu verstehen. Die Frage, um die sich diese Arbeit dreht, lautet: „Wie kann eine Anwendung, Studierende beim Lernen von Reduktionen unterstützen?“ Diese Anwendung ist eine Webanwendung, weshalb der größte Teil auch die Entwicklung des Frontends ist.

Wie der Name Reduktions-Trainer schon sagt, sollen mit ihm Reduktionen geübt werden können. Reduktion ist ein wichtiges Werkzeug für die Beweisführung in der Theoretischen Informatik. Mithilfe von Reduktionen kann zum Beispiel für Entscheidungsprobleme gezeigt werden, dass sie entscheidbar oder unentscheidbar sind. Wie Reduktion genau funktioniert und was Entscheidungsprobleme sind, wird im Kapitel 2 (Vorwissen) erläutert.

Im Folgenden noch eine kurze Beschreibung wie diese Ausarbeitung aufgebaut ist. Bevor es um den Reduktions-Trainer geht, werden im Kapitel 2 (Vorwissen), Definitionen aufgeführt und eine Intuition zu den Definitionen vermittelt. Nachdem das Vorwissen für den Reduktions-Trainer aufgefrischt wurde, wird in Kapitel 3 (High-Level Beschreibung des Reduktions-Trainers) ein Überblick über den Reduktions-Trainer und alle seine Komponenten gegeben. Dieses Kapitel erläutert vor allem den Aufbau und die geplante Funktionsweise des Reduktions-Trainers. Bei der Beschreibung der Umsetzung in Kapitel 4 (Umsetzung) wird nicht mehr der Aufbau behandelt, sondern wie

die einzelnen Komponenten tatsächlich als Webanwendung umgesetzt und welche Features implementiert wurden. Im letzten Kapitel, Kapitel 5 (Fazit) wird das Resultat dieser Arbeit zusammengefasst. Dabei wird auch erläutert, welche Funktionen den Reduktions-Trainer noch bereichern würden.

2 Vorwissen

In diesem Kapitel werden grundlegende formale Definitionen aufgeführt, die für Reduktionen und somit für den Reduktions-Trainer wichtig sind. Die hier aufgeführten Definitionen unterscheiden sich nicht oder nicht wesentlich von den Definitionen der Vorlesung "Berechenbarkeit und Komplexität", welche die Veranstaltung ist, wofür der Reduktions-Trainer begleitend eingesetzt werden soll. Im Folgenden werden unter anderem die Begriffe Entscheidbarkeit, Entscheidungsprobleme und die Reduktion selbst erläutert.

2.1 Berechenbarkeit

Zuerst wird der Begriff der Berechenbarkeit behandelt, denn ohne ein Verständnis von Berechenbarkeit lassen sich auch die anderen Begriffe nicht erklären. Berechenbarkeit wird in dieser Arbeit nur informell definiert, da es für die Zwecke dieser Arbeit nicht nötig ist, den Begriff formal zu erklären und zu weit vom eigentlichen Thema entfernt liegt.

Eine Funktion $f : \Sigma^* \rightarrow \Sigma^*$ heißt berechenbar, wenn es ein Python-Programm gibt, welches diese Funktion berechnet (Python [11]). Σ ist eine Menge von Zeichen, welche auch Alphabet genannt wird. Mit Σ^* ist jede beliebige Aneinanderreihung der Zeichen aus Σ gemeint. Eine Menge $M \subseteq \Sigma^*$ wird auch formale Sprache oder einfach nur Sprache genannt ([13], Seite 3). Außerdem wird jedes Element $w \in M$ Wort genannt. Da Berechenbarkeit nur über einen informellen Weg definiert wurde, wird ohne Beweis davon ausgegangen, dass es auch andere Berechnungsmodelle gibt, welche anstelle von Python eingesetzt werden könnten.

2.2 Entscheidbarkeit und Entscheidungsprobleme

Damit ist die Grundlage gelegt, um Entscheidbarkeit und Entscheidungsprobleme zu definieren. Entscheidbarkeit ist wie folgt definiert ([13], Seite 114).

Definition 2.1. *Eine Menge $A \subseteq \Sigma^*$ heißt entscheidbar, falls die charakteristische Funktion von A , nämlich $\chi_A : \Sigma^* \rightarrow \{0, 1\}$, berechenbar ist. Hierbei*

ist für alle $w \in \Sigma^*$:

$$\chi_A(w) = \begin{cases} 1, & w \in A \\ 0, & w \notin A \end{cases}$$

Eine Menge A mit der Frage, ob sie entscheidbar ist (Definition 2.1), also ob die Zugehörigkeit zur Menge berechenbar ist, wird hier Entscheidungsproblem (auch kurz Problem) genannt. Ein Entscheidungsproblem wird oft durch

<Name des Problems>

Gegeben: <Beschreibung der Instanz eines Problems>

Frage: <Frage bezüglich der gegebenen Instanz>

gegeben. Es folgt auch direkt noch ein konkretes Beispiel mit dem Grapherreichbarkeitsproblem, auch PATH genannt [12].

Grapherreichbarkeitsproblem (PATH)

Gegeben: Ein gerichteter Graph $G = (V, E)$ mit $s, t \in V$

Frage: Gibt es eine Pfad von s nach t ?

Hinter „Gegeben“ steht also, wie eine Instanz des Entscheidungsproblems aussieht. Beim Grapherreichbarkeitsproblem ist ein Graph gegeben, der als Tupel von Knoten V und Kanten E repräsentiert wird. s und t sind also Knoten im Graph G . Falls es nun in einem Graph G einen Pfad vom Knoten s zum Knoten t gibt, dann ist das eine Ja-Instanz des Grapherreichbarkeitsproblems. Bezogen auf die Definition 2.1 gibt die charakteristische Funktion in dem Fall 1 aus. Falls es keinen Pfad in dem Graph G von s nach t gibt, gibt die Funktion 0 aus und es ist eine Nein-Instanz.

Nun wurde noch nicht besprochen wie gezeigt werden kann, dass eine charakteristische Funktion eines Problems A berechenbar ist. Die charakteristische Funktion ist berechenbar wenn es einen Algorithmus gibt der sie berechnet. Dieser Algorithmus wird oft Entscheidungsverfahren genannt. Für den Reduktions-Trainer heißt das, wenn ein Problem entscheidbar ist, muss es auch ein Entscheidungsverfahren in Python gegeben, welches die charakteristische Funktion berechnet.

Interessant ist allerdings zu wissen, ob die charakteristische Funktion eines Problems überhaupt berechenbar ist. Ob eine Instanz eines Entscheidungsproblems eine Ja- oder eine Nein-Instanz ist, kann nur bestimmt werden, wenn die charakteristische Funktion berechenbar ist. Probleme, welche nicht entscheidbar sind, also jene für die die charakteristische Funktion nicht berechenbar ist, werden unentscheidbar genannt.

Unentscheidbare Probleme werden in dieser Arbeit nur sehr oberflächlich angesprochen, denn der Reduktions-Trainer ist in erster Linie und vor allem in der ersten Version, auf entscheidbare Probleme ausgelegt. Das ist auch nicht weiter verwunderlich, denn wie soeben vermutet, lässt sich nicht für jede Instanz eines unentscheidbaren Problems von einem Computer berechnen, ob es eine Ja- oder eine Nein-Instanz ist. Es ist also zunächst davon auszugehen, dass für Entscheidungsprobleme, die im Reduktions-Trainer vorkommen, die charakteristische Funktion berechenbar ist (Definition 2.1). Dass unentscheidbare Probleme existieren, wird hier als gegeben angenommen (Unentscheidbarkeit [13]).

Ein wichtiger Teil von Entscheidungsproblemen wurde noch nicht angesprochen, nämlich Codierung. Eine Instanz eines Entscheidungsproblems muss so codiert sein, dass ein Computer damit rechnen kann. Statt nun Codierung in diesem Kapitel formal zu definieren, was für den Reduktions-Trainer nicht direkt wichtig ist, wird die Codierung der Probleme im Reduktions-Trainer anhand von Beispielen im Kapitel 4 gezeigt.

2.3 Reduktion

Bevor Reduktion formal definiert wird, ist wichtig zu verstehen, was damit eigentlich erreicht werden soll und wie sich eine Reduktion vorgestellt werden kann.

Mithilfe von Reduktion kann gezeigt werden, dass ein Problem entscheidbar ist, indem es auf ein anderes Problem reduziert wird, von dem schon klar ist, dass es entscheidbar ist. Mit Reduktion kann auch noch weiteres gezeigt werden, aber eine grundlegende Intuition und Definition soll hier ausreichen.

Um eine Intuition einer Reduktion zu vermitteln, folgt ein Beispiel. Für eine Reduktion können sich zwei Entscheidungsprobleme A und B vorgestellt werden. Eine Reduktion funktioniert nun wie in Abbildung 1. Für Problem B gibt es bereits ein korrektes Entscheidungsverfahren. Dieses Entscheidungsverfahren ist ein Python-Programm, welches für jede korrekte Eingabe *true* ausgibt, wenn die Eingabe eine Ja-Instanz von B ist und *false*, wenn die Eingabe eine Nein-Instanz von B ist. Ein Entscheidungsverfahren muss kein Python-Programm sein, aber im Reduktions-Trainer sind Entscheidungsverfahren immer Python-Programme. Angenommen die Probleme A und B sind sich sehr ähnlich. Sie sind so ähnlich, dass direkt zu erkennen ist, dass durch eine kleine Abänderung der Eingabe in das Entscheidungsverfahren für A und dem anschließenden Aufruf des Entscheidungsverfahrens für B mit dieser ab-

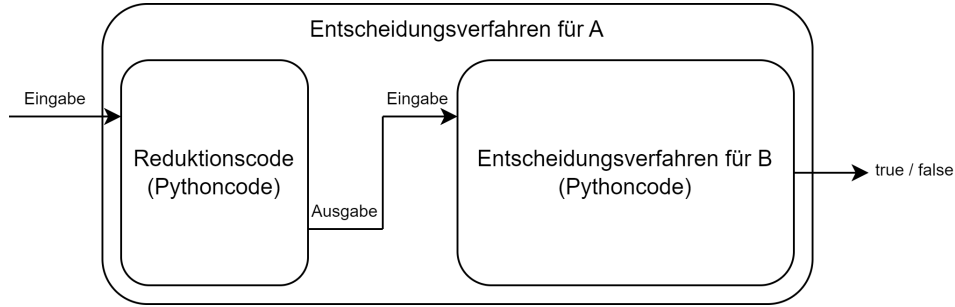


Abbildung 1: Intuition einer Reduktion

geänderten Eingabe, schon ein Entscheidungsverfahren für A entsteht. Diese Abänderung der Eingabe in das Entscheidungsverfahren für A ist in Abbildung 1 der Reduktionscode. Funktioniert nun der Reduktionscode korrekt für jede richtig codierte Eingabe, dann ist auch das Entscheidungsverfahren für A korrekt und damit wurde eine korrekte Reduktion von A auf B erstellt.

Formal lässt sich das Ganze wie folgt definieren ([13], Seite 120).

Definition 2.2. Seien $A \subseteq \Sigma^*$ und $B \subseteq \Gamma^*$ Sprachen. Dann heißt A auf B *reduzierbar* (symbolisch mit $A \leq B$ bezeichnet) falls es eine totale und berechenbare Funktion $f : \Sigma^* \rightarrow \Gamma^*$ gibt, so dass für alle $x \in \Sigma^*$ gilt:

$$x \in A \Leftrightarrow f(x) \in B.$$

In Definition 2.2 können mit Sprachen auch wieder Entscheidungsprobleme gemeint sein. Es folgt ein Lemma (und Beweis [13], Seite 120), welches eine wichtige Schlussfolgerung zieht, um für Probleme zu zeigen, dass sie entscheidbar sind.

Lemma 2.3. Falls $A \leq B$ und B entscheidbar ist, so ist auch A entscheidbar.

Beweis: Es gelte $A \leq B$ mittels Funktion f . Ferner sei χ_B berechenbar. Dann ist auch die Komposition beider Funktionen $\chi_B \circ f$ eine berechenbare Funktion. Es gilt

$$\chi_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases} = \begin{cases} 1, & f(x) \in B \\ 0, & f(x) \notin B \end{cases} = \chi_B(f(x))$$

Somit ist auch χ_A berechenbar, also A entscheidbar. ■

3 High-Level Beschreibung des Reduktions-Trainers

Im Folgenden wird das Konzept des Reduktions-Trainers erläutert. Genauer gesagt wird veranschaulicht wie der Trainer aussehen soll, welche Funktionen er haben soll und warum sich entschieden wurde ihn auf diese Weise zu kreieren.

Allgemein soll der Trainer den Studierenden helfen, vertrauter mit dem Erstellen eines Reduktions-Algorithmus zu werden. Studierende sollen also motiviert werden, sich mehr mit Reduktionen zu beschäftigen. Aus eigener Erfahrung kann ich sagen, dass es bei den Studierenden oft schon an den Grundlagen einer Reduktion scheitert. Durch die Möglichkeit, häufig zu testen und immer wieder Feedback zu bekommen, soll sowohl die Motivation als auch das Grundverständnis der Studierenden erhöht werden. Für die Dozenten bietet der Trainer eine neue Plattform, um den Studierenden viele Beispiele für die Anwendung von Reduktionen bereitzustellen.

3.1 Aufbau des Trainers

Direkt zu Beginn ist klar geworden, dass der Trainer eine Webanwendung werden soll, um für alle Studierenden gleichermaßen einfach zugänglich zu sein und außerdem einfach für zukünftige Erweiterungen aktualisierbar zu sein. Zudem gibt es auch schon andere Webanwendungen im zugehörigen Fachgebiet, welche das Lernen der Vorlesungsinhalte unterstützen sollen. Der volle Umfang des Reduktions-Trainers unterteilt sich in vier verschiedene Bereiche und der Kommunikation untereinander. Dazu gehört der Teil, welchen die Studierenden sehen, der Teil welchen die Dozenten sehen, der Teil welcher die Datenbank verwaltet, in der alle Übungen und Entscheidungsprobleme gespeichert sind und der Teil, welcher den Code einer Reduktion überprüft. Letzteres, also der Teil, der den Code der Reduktion überprüft, gehört nicht zu dieser Arbeit, ist aber essenziell notwendig, um den gesamten Reduktions-Trainer korrekt nutzen zu können. Dieser Teil wurde als eigenständige Bachelorarbeit von Kathrin Lehmann entwickelt [7]. Hier und im Folgenden wird dieser Teil als Reduktionsverifizierer bezeichnet und noch weitere Male erwähnt wenn es um die Reduktion mit eben diesem geht. In der Abbildung 2 ist die Kommunikation zwischen den einzelnen Bereichen bildlich dargestellt. Ein Bereich zeigt auf einen anderen, wenn von ihm aus die Kommunikation gestartet wird.

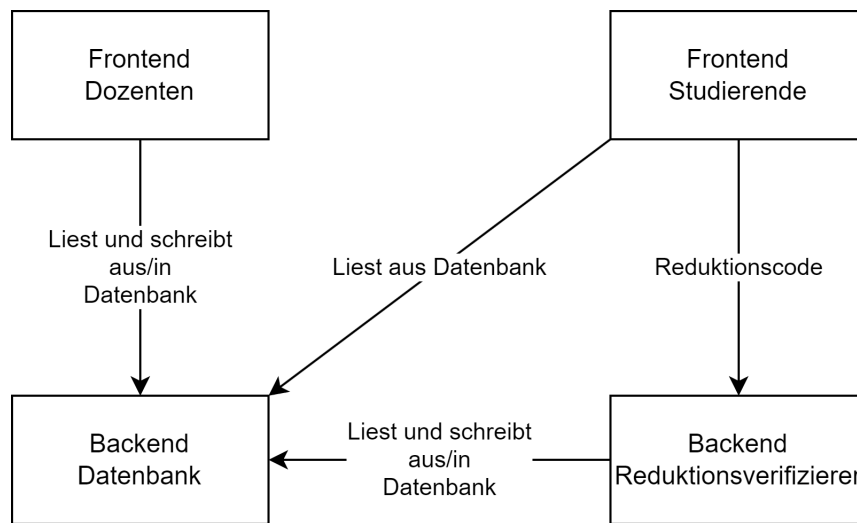


Abbildung 2: Übersicht Kommunikation

Die drei Komponenten „Frontend Dozenten“, „Frontend Studierende“ und „Backend Datenbank“ gehören alle zu dieser Arbeit und sie, sowie die Kommunikation zwischeneinander, werden im Folgenden noch ausführlicher erläutert.

Die meisten Informationen in der Datenbank werden vom Dozentenfrontend gegeben, aber alle Bereiche greifen auf die Informationen in der Datenbank zu. Das Studierenden-Frontend kommuniziert außerdem mit dem Reduktionsverifizierer, indem es Reduktionscode zu ihm schickt und eine Nachricht zurückbekommt, welche die Studierenden auf Fehler im Reduktionscode hinweisen soll. Zu guter Letzt schreibt auch der Reduktionsverifizierer in die Datenbank und zwar ausgewertete Testfälle, also Instanzen welche überprüft werden ob sie Ja- oder Nein-Instanzen von Entscheidungsproblemen sind.

Nun wird erläutert, was die einzelnen Bereiche bezwecken und wofür sie die Informationen, die sie kommunizieren, brauchen. Sinnvoll ist es mit dem Studierenden-Frontend zu beginnen, denn dann wird klar wozu alle anderen Bereiche notwendig sind.

3.2 Studierenden-Frontend

Das Studierenden-Frontend hat drei Ansichten (Abbildung 3), zwischen welchen gewechselt werden kann und mit denen sich im Folgenden befasst wird. Die Pfeile in der Abbildung sagen aus, wie sich zwischen den Ansichten hin

und her bewegt werden kann. Auf der Startseite stehen Übungen bereit, die

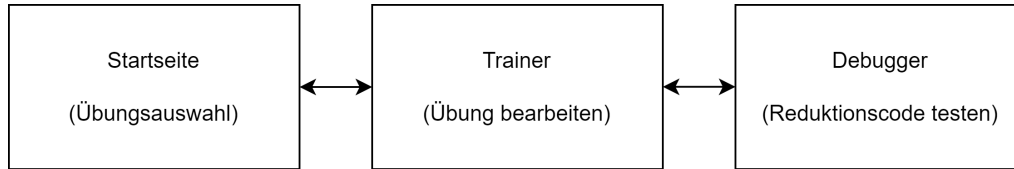


Abbildung 3: Übersicht Studierenden-Frontend

durch die Dozenten mit Informationen gefüllt wurden. Hier findet also eine Kommunikation mit der Datenbank statt, denn die Informationen jeder Übung wurden durch die Dozenten in der Datenbank hinterlegt. Eine Übung besteht aus zwei Entscheidungsproblemen, wobei eines auf das andere reduziert werden soll. Die Studierenden können dann auf der Startseite auswählen, welche Übung sie bearbeiten möchten. Zu jeder Übung steht deshalb auch der Name der Übung, die Namen der Entscheidungsprobleme und ob die Reduktion in Polynomialzeit durchführbar sein soll. Zu diesen Informationen muss es den Studierenden natürlich noch möglich sein, die Übung zu bearbeiten. Mit einem, zu einer Übung zugehörigen, Button kann sich der/die Studierende zur Ansicht des Trainers bewegen.

Im Trainer können die Studierenden die ausgewählte Übung bearbeiten. Dafür kann der Code, der die Reduktion lösen soll, eingegeben werden. Zudem gibt es wieder Informationen über die Entscheidungsprobleme, welche aus der Datenbank geholt werden. Nachdem Reduktionscode eingegeben wurde, können die Studierenden durch das Klicken auf einen Button den Code an den Reduktionsverifizierer senden, der diesen kontrolliert und eine Antwort zurücksendet. Wie der Reduktionsverifizierer genau funktioniert, kann in der Bachelorarbeit von Kathrin Lehmann nachgelesen werden [7]. Als Antwort bekommt der/die Studierende dann vom Reduktionsverifizierer eine Nachricht, ob Fehler im Reduktionscode ausfindig gemacht wurden oder ob die Reduktion korrekt zu sein scheint. An dieser Stelle hat der/die Studierende schon die Möglichkeit, durch die Nachrichten des Reduktionsverifizierers dazuzulernen, um den/der Studierenden aber noch mehr Möglichkeiten zu geben, sich mit der Reduktion auseinanderzusetzen, gibt es noch den Debugger.

Im sogenannten Debugger kann der Reduktionscode mit selbstbestimmten Eingaben getestet werden. In Abbildung 4 ist der Debugger verallgemeinert dargestellt. Direkt auf den ersten Blick ist zu erkennen, dass der Debugger genau so aufgebaut ist, wie sich eine Reduktion vorgestellt werden kann. Er

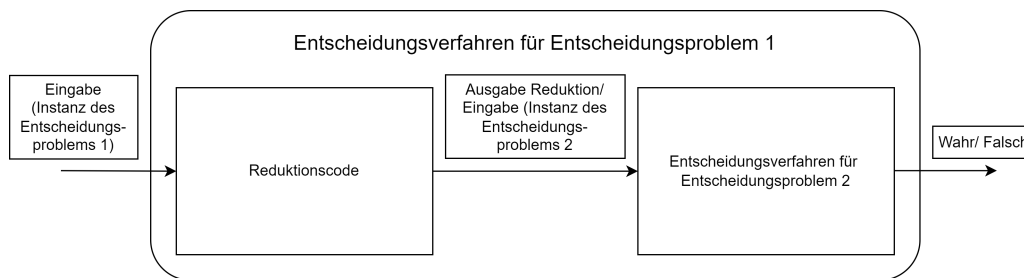


Abbildung 4: Übersicht Debugger

sieht der Intuition einer Reduktion, wie sie in Abbildung 1 schon zu sehen war, sehr ähnlich und leitet sich direkt aus dem Lemma 2.3 ab. Denn im Lemma steht „ist B entscheidbar“, was soviel heißt, dass es ein berechenbares Entscheidungsverfahren für B gibt. Das ist in Abbildung 4 das Entscheidungsverfahren für Entscheidungsproblem 2. „und $A \leq B$ “, was heißt, dass es eine totale und berechenbare Funktion gibt, wie sie in Definition 2.2 definiert ist. Das ist in Abbildung 4 der Reduktionscode. „so ist auch A entscheidbar“, was heißt, dass es dann ein Entscheidungsverfahren für A gibt, welches in Abbildung 4 das Entscheidungsverfahren für Entscheidungsproblem 1 ist.

Der Debugger kann nun von den Studierenden genutzt werden, indem der Reduktionscode und die Eingabe, also eine Instanz des ersten Problems, eingegeben wird. Die Ausgabe des Reduktionscodes, welche natürlicherweise auch die Eingabe in das zweite Entscheidungsverfahren ist, ergibt sich durch das Ausführen des Reduktionscodes mit der eingegebenen Instanz des ersten Entscheidungsproblems. Um eine Auswertung, also die Ausgabe des Reduktionscodes und die Ausgabe des gesamten Entscheidungsverfahrens (Entscheidungsverfahren für Entscheidungsproblem 1 in Abbildung 4), zu erhalten, werden die Eingabe, der Reduktionscode und das Entscheidungsverfahren des zweiten Entscheidungsproblems (oder zumindest der Name, um sich das Entscheidungsverfahren aus der Datenbank zu holen) an den Reduktionsverifizierer gesendet. Dieser schickt dann als Auswertung, wenn die Eingaben des/der Studierenden fehlerfrei waren, die Ausgabe des Reduktionscodes und die Ausgabe des Entscheidungsverfahrens zurück, um sie dem/der Studierenden anzeigen zu können. Die Studierenden können also durch verschiedene Eingaben testen, ob ihr Reduktionscode so funktioniert wie sie es erwarten.

Damit wurde ein Überblick über die Funktionsweise des Frontends der Studierenden gegeben. Was noch nicht beschrieben wurde, ist das Frontend der

Dozenten, welches für viele Informationen, im gerade beschriebenen Frontend für Studierende, notwendig ist.

3.3 Dozenten-Frontend

Ganz allgemein soll das Dozenten-Frontend bestimmte Informationen, die die Dozenten bereitstellen, an die Studierenden geeignet weitergeben. Dazu braucht es eine Anwendung, die die Dozenten beim Eingeben und übermitteln der Informationen unterstützt. Das Dozenten-Frontend ist anschaulich in Abbildung 5 zu sehen. Es besteht aus mehreren verschiedenen Ansich-

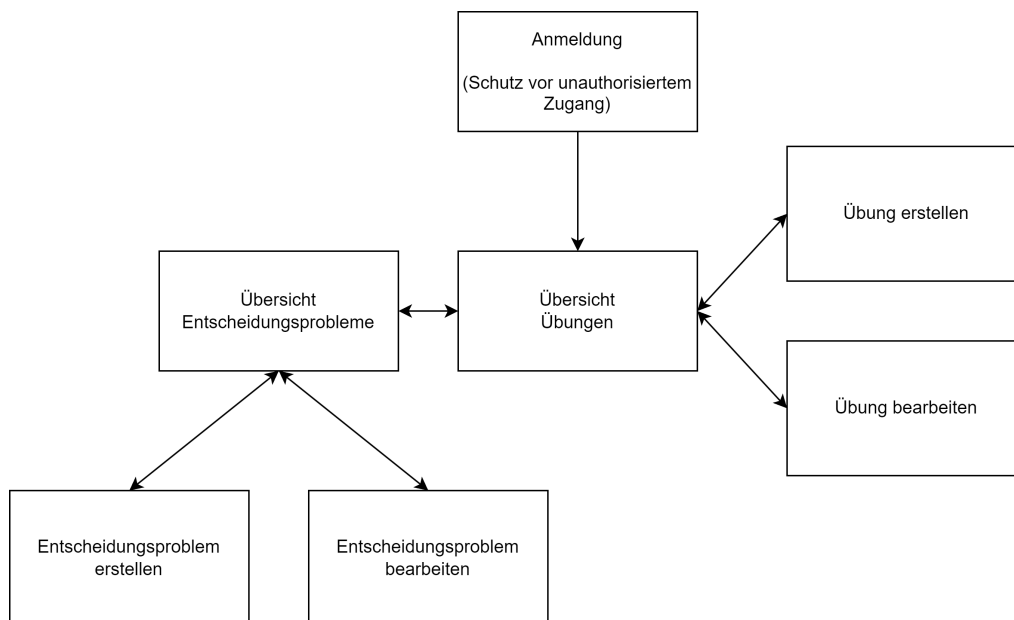


Abbildung 5: Übersicht Dozenten-Frontend

ten zwischen denen, wie die Pfeile angeben, hin und her gewechselt werden kann. Im Gegensatz zum Frontend für die Studierenden muss das Frontend für die Dozenten vor unbefugtem Zugriff geschützt werden. Studierende sollen natürlich nicht selbst die Übungen und Entscheidungsprobleme anpassen können. Deshalb gibt es dafür extra eine Ansicht, in der ein Passwort abgefragt wird. Andere Methoden, um sich als Dozent zu verifizieren, wären auch denkbar. Das Passwort wird in Kommunikation mit dem Datenbank Backend überprüft, liegt aber nicht direkt in der Datenbank.

Die erste Ansicht nach der Anmeldung ist die Übersicht über die Übungen. Wie in Abbildung 2 zu sehen, wird aus dem Frontend der Dozenten

lesend und schreibend auf die Datenbank zugegriffen. In der Übersicht über die Übungen wird nun also mit dem Backend der Datenbank kommuniziert. Hier sind alle Details zu den Übungen, die in der Datenbank vorhanden sind, einsehbar. Jede Übung repräsentiert eine Reduktion, angezeigt werden also Informationen wie Namen der Entscheidungsprobleme, die aufeinander reduziert werden, Beschreibungen zu den Entscheidungsproblemen und Name der Übung. Von hier aus gibt es verschiedene Möglichkeiten, was die Dozenten machen können. Zum einen hat der Dozent die Möglichkeit, Übungen zu löschen, was sich hier anbietet, da alle Übungen eingesehen werden können. Außerdem kann der Dozent durch das Anklicken eines zugehörigen Buttons die Ansicht wechseln (siehe Abbildung 5), um eine Übung zu erstellen, eine Übung zu bearbeiten oder aber zur Übersicht über die Entscheidungsprobleme zu kommen.

In der Ansicht Übung erstellen kann der Dozent eine neue Übung anlegen. Dazu müssen alle Informationen, die eine Übung braucht, um für die Studierenden angemessen präsentiert werden zu können, eingegeben werden. Da eine Übung eine Reduktion repräsentiert, enthält sie zwei Entscheidungsprobleme. Die Entscheidungsprobleme können vom Dozenten eingegeben oder falls vorhanden, aus der Datenbank geladen werden. Entscheidungsprobleme können einzeln geladen werden, denn zusätzlich dazu dass sie bei jeder neu erstellten Übung gespeichert werden, können Entscheidungsprobleme auch einzeln erstellt werden, siehe Abbildung 5. Weitere Details dazu, welche Informationen genau nötig und sinnvoll für eine Übung sind, gibt es im Kapitel 4. Wenn alle Information eingegeben wurden, wird wieder schreibend mit dem Backend, welches Zugriffe auf die Datenbank gewährt, bzw. mit der Datenbank kommuniziert und die Übung in der Datenbank gespeichert. Nach dem Speichern der Übung in der Datenbank oder nach dem Abbrechen des Vorgangs wird wieder zur Übersicht über die Übungen gewechselt.

Die Ansicht „Übung bearbeiten“ verhält sich ganz ähnlich wie die Ansicht „Übung erstellen“. Hier wird allerdings gleich zu Beginn mit der Datenbank bzw. dem Datenbank Backend kommuniziert. Die in der Übungen Übersicht zum Bearbeiten ausgewählte Übung wird hier aus der Datenbank geladen. Alternativ wäre es auch möglich, alle Informationen über die Übung aus der vorherigen Ansichten zu übergeben. Nachdem die Änderungen von den Dozenten bestätigt wurde, wird wieder mit dem Datenbank Backend kommuniziert und die Übung überschrieben. Im Anschluss daran oder beim Abbrechen des Vorgangs wird auch hier zurück zur Übersicht der Übungen gewechselt.

Die dritte Ansicht zu der die Dozenten in der Übersicht der Übungen wechseln können ist die Übersicht über die Entscheidungsprobleme. Diese Übersicht verhält sich im Grunde genauso wie die Übersicht über die Übungen, nur dass diese Entscheidungsprobleme behandelt. Dies ist vor allem sinnvoll, da jede Übung hauptsächlich aus zwei Entscheidungsproblemen besteht und die Dozenten so nicht immer eine Reduktion vorbereiten müssen, wenn sie ein ansprechendes Entscheidungsproblem im Sinn haben. In dieser Übersicht gibt es also auch die Möglichkeit, Entscheidungsprobleme aus der Datenbank zu löschen, zu den Ansichten Problem erstellen und Problem bearbeiten zu wechseln (Abbildung 5) und wieder zurück zur Übersicht über die Übungen zu wechseln. Die Funktionen, sowie die zwei Ansichten um Entscheidungsprobleme zu erstellen und zu bearbeiten verhalten sich also analog dazu, Übungen zu erstellen und zu bearbeiten, was zuvor behandelt wurde.

3.4 Datenbank Backend

Das Backend, welches die Datenbank verwaltet, macht dieses wie in Abbildung 6. An das Datenbank Backend kann eine Anfrage gestellt werden.

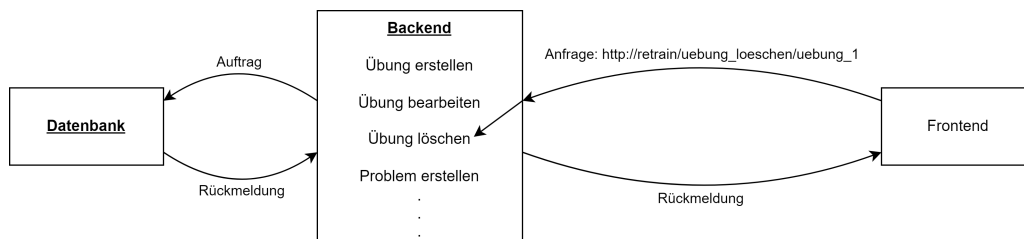


Abbildung 6: Übersicht Datenbank Backend

Diese kann zum Beispiel aus einem der zuvor beschriebenen Frontends kommen, wie das in der Abbildung 6 ganz rechts angedeutet ist. Das Backend ist über verschiedene Adressen oder auch Routen genannt erreichbar, welche die Information mitgeben, was in der Datenbank geschehen soll. Soll also eine Übung gelöscht werden, wird mit dem Backend über die zugehörige Route kommuniziert und es wird der Übungsname der Übung, die gelöscht werden soll, hinzugefügt. Dort wird dann eine Funktion ausgeführt, die die Übung in der Datenbank löscht. Die Datenbank gibt dem Backend dann eine Bestätigung zurück, wenn der Vorgang erfolgreich war oder einen Fehler wenn nicht. Schlussendlich schickt das Backend dann eine Meldung an die anfragende Quelle zurück. Welche Funktionen das Backend alles hat, wie die Datenbank angesprochen werden kann und welches Datenbankmanagementsystem genutzt wird, wird im Kapitel 4 genauer erläutert.

Auch der Reduktionsverifizierer kommuniziert mit der Datenbank bzw. dem zugehörigen Backend. Um gut zu funktionieren, benutzt der Reduktionsverifizierer Testfälle von Entscheidungsproblemen, welche Instanzen von genau diesen Entscheidungsproblemen sind. In der Datenbank müssen also zu jedem Entscheidungsproblem, welches dort hinterlegt ist, auch immer Testfälle beigefügt werden. Die Testfälle werden im Dozenten-Frontend erstellt, aber vom Reduktionverifizierer ausgewertet. Ob und wie die Testfälle ausgewertet wurden, ist Teil jedes Testfalls in der Datenbank.

4 Umsetzung

Das Kapitel Umsetzung behandelt, die Frontend-Entwicklung des Reduktions-Trainers. Folgende Technologien wurden dafür benutzt. Beide Frontends wurden mit der Programmiersprache TypeScript [14] bzw. JavaScript [4] programmiert. Dazu wurde das Framework Angular [1] benutzt und ein Großteil des Interfaces wurde mit der für Angular ausgelegten UI Bibliothek PrimeNG [10] erstellt.

In den kommenden Kapiteln werden geplante Inhalte der Anwendung, in Form von Mockups, mit der tatsächlichen Umsetzung, mithilfe von Screenshots, verglichen. Als Erstes wird das Frontend für die Studierenden erläutert.

4.1 Benutzeroberfläche für Studierende

Dieses Kapitel befasst sich mit dem für die Studierenden sichtbaren Teil des Reduktions-Trainers. Ein Überblick darüber, wie das Frontend für die Studierenden aussieht, wurde schon im vorherigen Kapitel gegeben (siehe Abbildung 3). Hier wird sich also nur noch damit befasst wie es schlussendlich umgesetzt wurde.

Das Erste, was die Studierenden sehen, ist die Übungsansicht.

Was im gesamten Frontend, also sowohl für die Studierenden als auch für die Dozenten, immer zu sehen ist, ist die Leiste am oberen Rand in der die Überschrift steht. Die Leiste wurde genau wie geplant umgesetzt, wie schon im Mockup (Abbildung 7) und im Screenshot (Abbildung 8 zur Übungsauswahl zu sehen ist). Klickt der/die Nutzer/in auf das Fragezeichen-Symbol, welches etwas weniger auffällig als im Mockup ist (rote Markierung Nummer 2, Abbildung 9), wird er/sie zum Impressum weitergeleitet. Die Impressums-Seite (Abbildung 11) wurde im Vergleich zum Mockup (Abbildung 10) auf die Aufführung der Entwickler des Reduktions-Trainers reduziert. Zurück zur

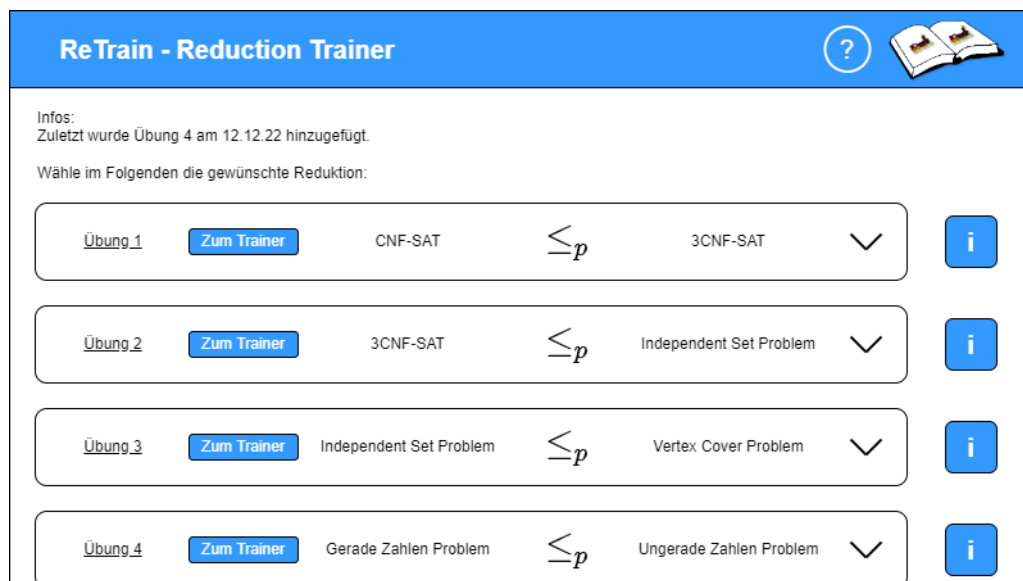


Abbildung 7: Mockup Übungsauswahlseite Studierenden-Frontend

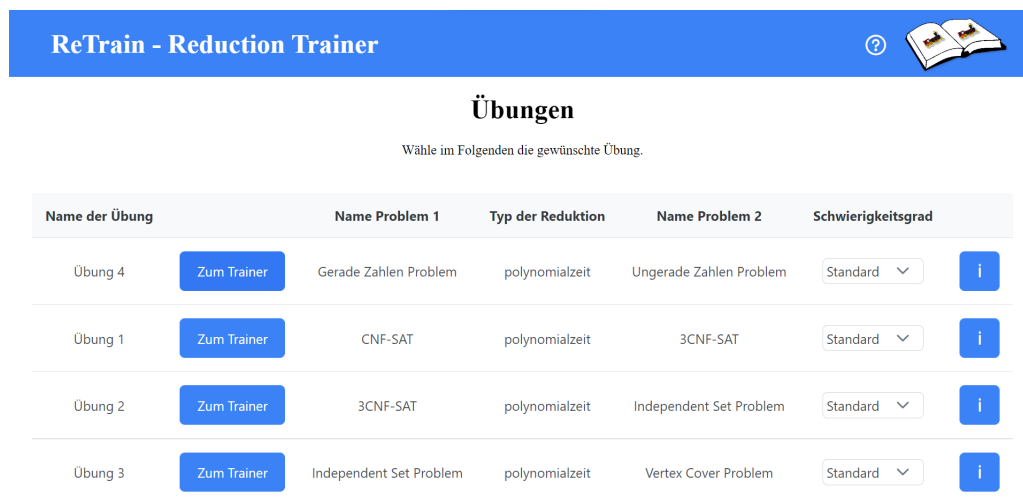


Abbildung 8: Screenshot Übungsauswahlseite Studierenden-Frontend

Übungsauswahl können die Studierenden, indem sie auf die Leiste am oberen Rand klicken (rote Markierung Nummer 1, Abbildung 9). Mit einem Klick auf die Leiste kann auch aus jeder anderen Ansicht wieder auf die Startseite gewechselt werden. Die Startseite ist im Frontend für die Studierenden die Übungsauswahlansicht, im Frontend für die Dozenten ist es die Login-Seite.

ReTrain - Reduction Trainer

2

?

1

Übungen

3

Wähle im Folgenden die gewünschte Übung.

Name der Übung		Name Problem 1	Typ der Reduktion	Name Problem 2	Schwierigkeitsgrad			
Übung 4	6	<div>Zum Trainer</div>	Gerade Zahlen Problem	polynomialzeit	Ungerade Zahlen Problem	4	<div>Standard</div>	<div>i</div> 5
Übung 1		<div>Zum Trainer</div>	CNF-SAT	polynomialzeit	3CNF-SAT		<div>Standard</div>	<div>i</div>
Übung 2		<div>Zum Trainer</div>	3CNF-SAT	polynomialzeit	Independent Set Problem		<div>Standard</div>	<div>i</div>
Übung 3		<div>Zum Trainer</div>	Independent Set Problem	polynomialzeit	Vertex Cover Problem		<div>Standard</div>	<div>i</div>

Abbildung 9: Screenshot Übungsauswahlseite Studierenden-Frontend mit Markierungen

ReTrain - Reduction Trainer

?

1

Der Reduction Trainer wurde zur Begleitung der Vorlesung entwickelt, Definitionen und Beispiele zur Reduktion oder Entscheidungsproblemen findest du also in den Vorlesungsfolien. Hast du allgemein Fragen zu Reduktionen, dann richte diese bitte an die Betreuer der derzeitigen Vorlesung.

Impressum:

Lizenzen: MIT

Abbildung 10: Mockup Impressum

Wie auch in jeder anderen Ansicht des Reduktions-Trainers (ausgenommen, die Login-Ansicht) gibt es eine Überschrift und direkt darunter eine Kurzbeschreibung der Ansicht (rote Markierung Nummer 3, Abbildung 9). Diese dienen zur Orientierung. Im Mockup (Abbildung 7) steht lediglich eine

Abbildung 11: Screenshot Impressum

Beschreibung, welche in der Anwendung ohne Überschrift allerdings schnell übersehen wird.

Alles unter der Beschreibung (Abb. 7) wurde in einer Tabelle zusammengefasst. Die Funktionen, welche auch schon bei der Erstellung des Mockups geplant wurden, wurden im Grunde auch so in der Webanwendung umgesetzt (Abb. 8). Zur Übersicht hat die Tabelle Überschriften für jede Spalte, bis auf die in der die Buttons stehen. Genau wie auch schon im Mockup bekommt jede Übung eine eigene Zeile und die Informationen, die es an dieser Stelle über die Übung gibt, sind die Gleichen, auch wenn sie etwas verändert dargestellt sind. Statt die gesamte Zeile der Übung auszuklappen wie im Mockup, hat jede Übung ein kleines Drop-down-Menü, in dem die Schwierigkeit ausgewählt werden kann (rote Markierung Nummer 4, Abb. 9). Durch Schwierigkeitsgrade soll die Bearbeitung einer Übung vereinfacht werden können. Wird ein einfacherer Schwierigkeitsgrad als Standard ausgewählt, dann wird für die Studierenden Code für die Reduktion vorgegeben. Die Dozenten müssen dafür vorher einen neuen Schwierigkeitsgrad erstellt haben, indem sie Code angeben, der den Studierenden beim Lösen der Übung hilft. Wie genau das für die Dozenten möglich ist folgt noch in kommenden Kapiteln. Ein Schwierigkeitsgrad könnte einfach nur ein Kommentar sein, welcher einen Tipp enthält oder auch Code für die Reduktion vorgeben, wie in Abbildung 14.

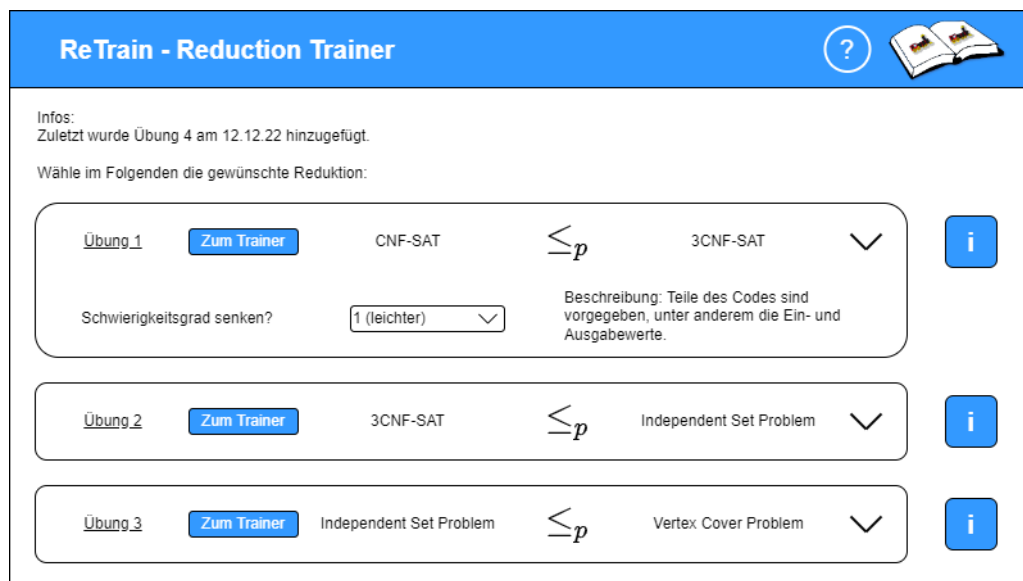


Abbildung 12: Mockup Schwierigkeitsgrad

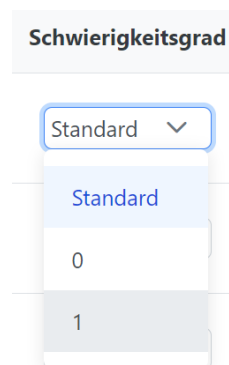


Abbildung 13: Screenshot Schwierigkeitsgrad

```

1 def even_into_uneven(inp_tuple):
2     number_list = list(inp_tuple[0])
3     # Hier fehlt noch Code
4
5     return number_list,

```

Abbildung 14: Screenshot vorgegebener Code für eine Reduktion

Bei der Erstellung des Mockups wurde noch davon ausgegangen, dass jede Schwierigkeit auch eine Beschreibung und einen eigenen Namen braucht,

welche von Dozenten eingegeben werden müssten. Davon wurde bei der Umsetzung abgesehen, um die Dozenten bei der Eingabe etwas zu entlasten. Stattdessen gibt es für jeden Schwierigkeitsgrad, den Dozenten anlegen, eine neue Zahl. Der Schwierigkeitsgrad Standard sagt aus, dass es keine Hilfestellung für diese Übung gibt. Jeder weitere Schwierigkeitsgrad, der von Dozenten angelegt wird, soll die Lösung der Übung weiter vereinfachen. Gibt es also nur den Schwierigkeitsgrad Standard, wurde keine Schwierigkeit angelegt. Gibt es stattdessen noch die Schwierigkeitsgrade 0 und 1 wurden zwei Schwierigkeitsgrade zu dieser Übung angelegt.

Der Infobutton (rote Markierung Nummer 5, Abb. 9) ist in jeder Zeile zur Übung platziert, genau wie auch im Mockup. Beim Anklicken des Infobuttons werden Beschreibungen zu den Problemen eingeblendet, wie in Abbildung 16 zu sehen ist und im Vergleich dazu war es auch schon im Mockup in Abbildung 15 so geplant. Wie in Abbildung 16 zu sehen ist, werden die

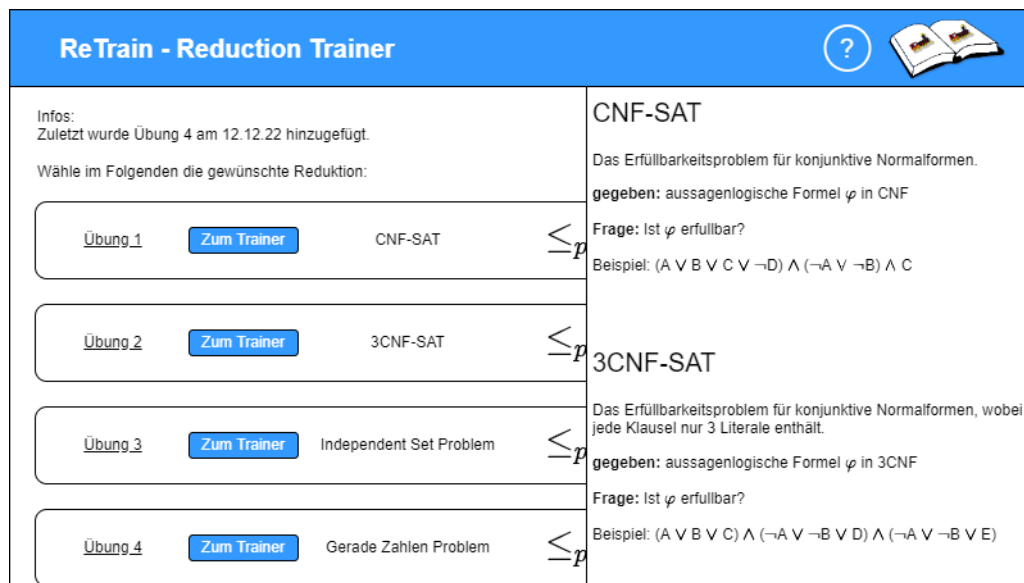


Abbildung 15: Mockup Infobutton geklickt

Beschreibungen von links statt von rechts, wie im Mockup, eingeblendet. Das wurde deshalb so umgesetzt, weil dann zu sehen ist welcher der Infobuttons geklickt wurde und zu welcher Zeile die Beschreibungen gehören. Ansonsten unterscheidet sich das Mockup von der Umsetzung noch im Stil der Beschreibung, welche allerdings im Frontend für die Dozenten verändert wurde und deshalb später noch in einem späteren Kapitel zur Sprache kommt.

Gerade Zahlen Problem

Liste L mit natürlichen Zahlen
Ist jede Zahl k in L gerade?
Beispiel: [2, 4, 12, 18]

Ungerade Zahlen Problem

Liste L mit natürlichen Zahlen
Ist jede Zahl k in L ungerade?
Beispiel: [3, 7, 17, 21]

on Trainer

Übungen

Wähle im Folgenden die gewünschte Übung.

Name Problem 1	Typ der Reduktion	Name Problem 2	Schwierigkeitsgrad	
Gerade Zahlen Problem	polynomialzeit	Ungerade Zahlen Problem	Standard	i
CNF-SAT	polynomialzeit	3CNF-SAT	Standard	i
3CNF-SAT	polynomialzeit	Independent Set Problem	Standard	i
Independent Set Problem	polynomialzeit	Vertex Cover Problem	Standard	i

Abbildung 16: Screenshot Infobutton geklickt

Der andere Button, welcher in jeder Zeile der Tabelle steht, ist der „Zum Trainer“-Button (rote Markierung Nummer 6, Abb. 9). Mit diesem Button wird die Ansicht zum Trainer gewechselt und die zum Button zugehörige Übung geladen. Wird nun also auf den „Zum Trainer“-Button der Übung 4 geklickt, kann die Übung 4 in der Traineransicht (Abb. 19) bearbeitet werden. Wie schon in Kapitel 3 besprochen, kann in der Traineransicht die ausgewählte Reduktion bearbeitet und vom Reduktionsverifizierer überprüft werden. Die Traineransicht (Abbildung 19) wurde dabei fast genauso wie in den Mockups (Abbildungen 17, 18) umgesetzt. Abgesehen von der Überschrift und der Unterüberschrift, dessen Änderung schon in der Übungsauswahlansicht besprochen wurde und auf allen Ansichten gleich ist, sind die Änderungen gering. Wie schon zuvor werden die einzelnen Komponenten der Ansicht, sowie Funktionen und Änderungen zum Mockup wieder mithilfe von Markierungen in der Ansicht besprochen (Abbildung 20). Mit dem „Zurück zur Übungsansicht“-Button (rote Markierung Nummer 1, Abb. 20) können sich die Studierenden, zurück zur Übungsauswahlansicht (Abb. 8) begeben, deren Funktionen zuvor beschrieben wurden.

Mittig im oberen Teil der Ansicht (rote Markierung Nummer 2, Abb. 20) steht welches Problem auf welches andere Problem reduziert werden soll. Genauso wie es auch schon in den Mockups zu sehen ist (Abb. 17, 18). Zusätzlich ist durch das Zeichen der Reduktion, zwischen den beiden Namen

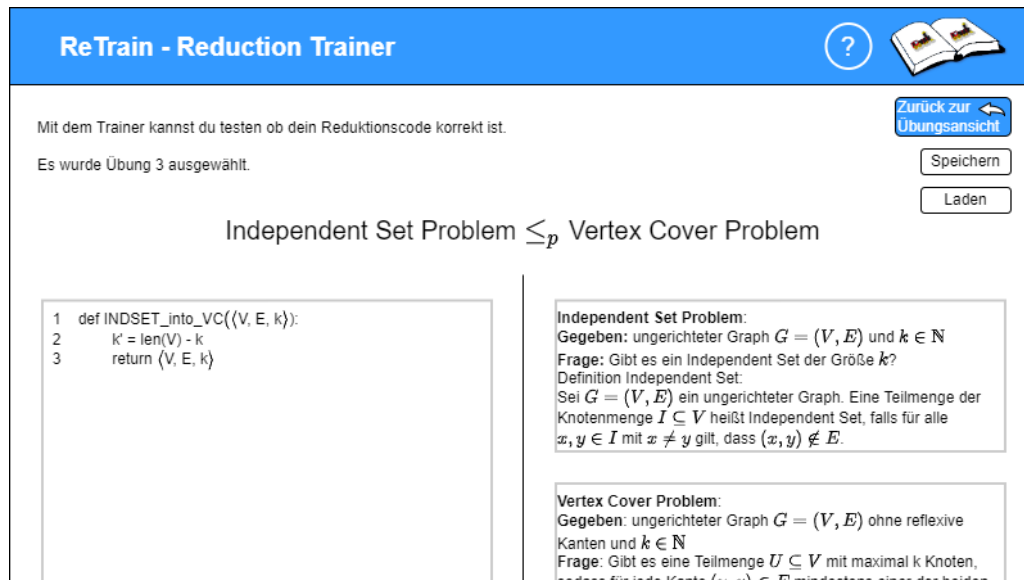


Abbildung 17: Mockup Trainer oben

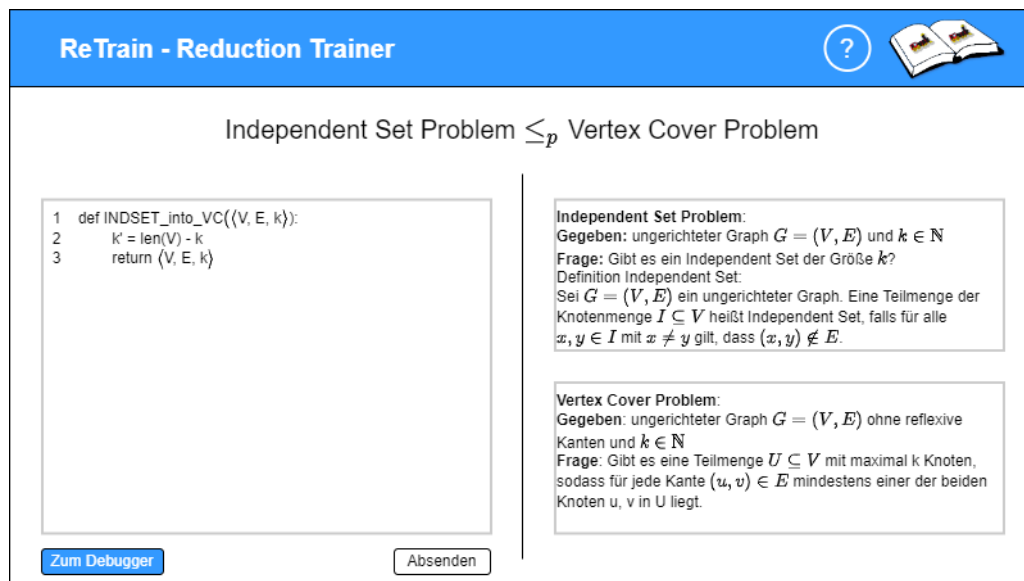


Abbildung 18: Mockup Trainer unten

der Probleme zu erkennen, ob die Reduktion in Polynomialzeit oder nicht in Polynomialzeit stattfinden soll. Zur Erinnerung, es steht \leq_p zwischen den Problemen, wenn die Reduktion in Polynomialzeit durchführbar sein soll und es steht \leq zwischen den Problemen, wenn die Reduktion nicht in Polynomialzeit durchführbar sein muss.

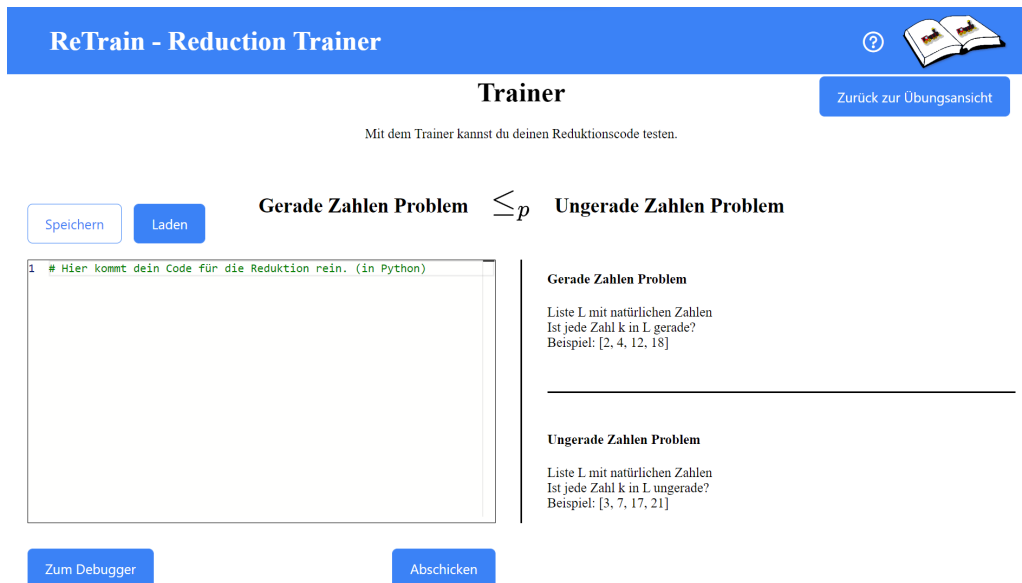


Abbildung 19: Screenshot Trainer



Abbildung 20: Screenshot Trainer mit Markierungen

Einen großen Teil der Ansicht nehmen die Beschreibungen (rote Markierung Nummer 3, Abb. 20) der Probleme ein. Diese dienen sowohl als Erinnerung und Hilfestellung beim Lösen der Reduktion als auch um die richtige Codierung zu erfahren. Bei der Beschreibung liegt die Verantwortung bei den

Dozenten, alle nötigen Informationen an die Studierenden weiterzugeben. Wie schon in Kapitel 2.2 besprochen, kann ein Entscheidungsproblem durch Angabe eines „gegeben“-Teils und eines „Frage“-Teils beschrieben werden. Besonders wichtig für die Codierung ist, was für das Problem gegeben ist. Dafür ist wichtig zu verstehen was im Editor (rote Markierung Nummer 4, Abb. 20), links neben den Beschreibungen stehen soll.

Im Editor kann mit der Programmiersprache Python [11] ein Algorithmus für die Reduktion angegeben werden. Ein Hinweis dafür steht als Kommentar im Editor. Wie auch bei einer Reduktion auf Papier wird eine Reduktionsfunktion angegeben welche hier nun wie folgt aufgebaut sein soll.

```
def <Name der Funktion>(<Name des Eingabeparameters >):
    <Code>
    return <Rueckgabewert als Tupel>
```

Code Abschnitt 1: Aufbau Reduktionsfunktion

Wie im Code Abschnitt 1 zu sehen ist, wird eine Funktion in Python mit „def“ angegeben. Darauf folgt nach einem Leerzeichen der Name der Funktion, im Code Abschnitt 1 steht dafür ein Platzhalter in spitzen Klammern. Im Code Abschnitt 1 ist alles in spitzen Klammern, inklusive der spitzen Klammern, Platzhalter. Interessant ist nun was als Parameter in den Klammern der Funktionsdefinition steht. Es gibt immer nur einen Parameter, welcher ein Tupel ist, und für den ein beliebiger Name gewählt werden kann. Das Eingabe-Tupel enthält dann das, was für ein Problem gegeben ist. Konkret ist das für das Gerade Zahlen Problem eine Liste mit natürlichen Zahlen. Um nun auf die Liste zuzugreifen, muss also auf das erste Element des Eingabe-Tupels zugegriffen werden. In Python könnte die Liste des Tupels in eine Variable gesteckt werden, um damit weiter zu arbeiten. Das könnte (ohne Einrückung) wie folgt aussehen.

```
number_list = list(inp_tuple[0])
```

Dabei ist `inp_tuple` das Eingabe-Tupel. Da das zweite Problem auch immer ein Tupel als Eingabeparameter entgegennimmt, muss auch der Rückgabewert der Funktion ein Tupel sein, welches das beinhaltet, was für das zweite Problem gegeben ist. Konkret könnte das (ohne Einrückung) wie folgt aussehen.

```
return number_list ,
```

Dabei könnte `number_list` eine veränderte Liste aus dem Eingabe-Tupel sein und das Komma am Ende sorgt dafür, dass die Ausgabe als Tupel interpretiert wird.

Der wichtigste Grund, warum ein Tupel als Eingabeparameter für die Reduktionsfunktion gewählt wurde, der alle Parameter zusammenfasst, ist, dass die Reihenfolge der Parameter nicht beliebig verändert werden soll. Das erleichtert die automatische Korrektur der Reduktionsfunktion.

Bei der automatischen Korrektur, die vom Reduktionsverifizierer durchgeführt wird, dürfen manche Schlüsselwörter im Reduktionscode nicht verwendet werden. Aus diesem Grund werden die Studierenden direkt bei der Eingabe in den Editor darauf hingewiesen. In Abbildung 21 ist zu sehen, dass verbotene Schlüsselwörter rot unterstrichen werden. Zusätzlich wird der

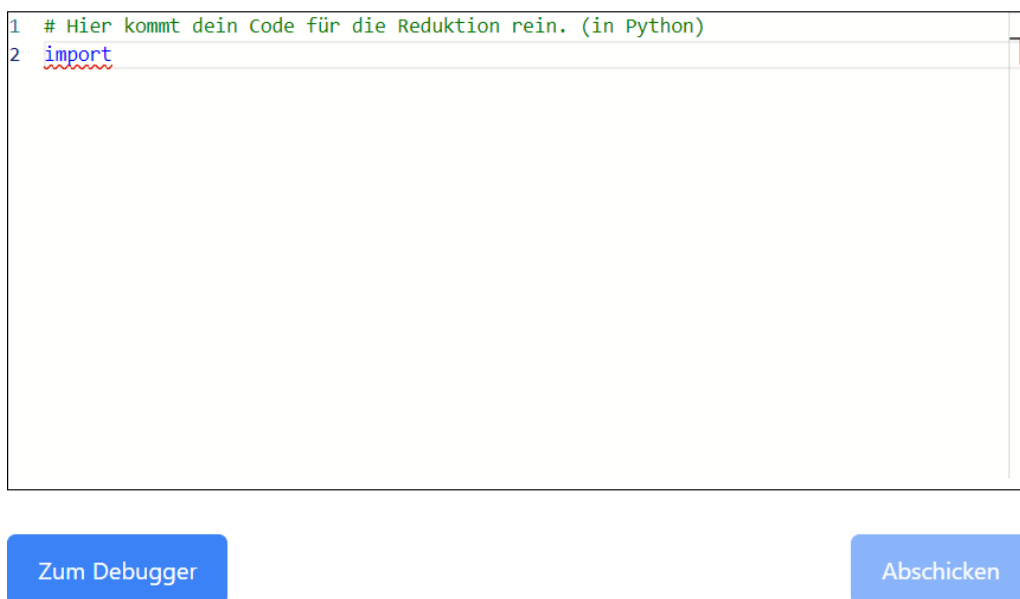


Abbildung 21: Screenshot Markierung verbotener Schlüsselwörter

„Abschicken“-Button genau dann deaktiviert, wenn mindestens ein solches Schlüsselwort im Editor steht. Wird der Mauszeiger über ein rot unterstrichen Schlüsselwort gehalten, wird außerdem eine Fehlermeldung angezeigt, wie in Abbildung 22 zu sehen ist. Die Maßnahme, bestimmte Schlüsselwörter

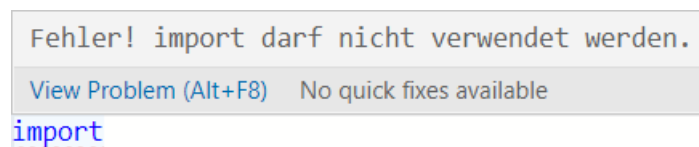


Abbildung 22: Screenshot Fehlermeldung verbotenes Schlüsselwort

ter für die Studierenden im Frontend zu verbieten, ist dafür da, um deutlich

darauf hinzuweisen, diese Wörter nicht zu verwenden. Allerdings ist sie kein vollständiger Schutz für den Reduktionsverifizierer vor Code, welcher die Integrität des Systems beeinträchtigen könnte. Deshalb müssen dort weitere Vorsichtsmaßnahmen eingehalten werden.

Mit den „Speichen“- und „Laden“-Buttons (rote Markierung Nummer 5, Abb. 20), direkt über dem Editor, kann der Reduktionscode in eine Datei gespeichert oder aus einer Datei geladen werden. Dabei wird beim Speichern lokal auf dem Computer eines/einer Studierenden eine TXT-Datei angelegt, wobei der Speicherort und Name, wenn vom Betriebssystem unterstützt, noch verändert werden kann. Beim Laden muss dann lokal eine TXT-Datei ausgewählt werden, deren Inhalt in den Editor geladen wird. Beim Speichern wird auch ein Name der Datei vorgeschlagen, welcher die Namen der Probleme enthält. Für die Übung in Abbildung 19 würde beim Speichern als Name „Gerade Zahlen Problem auf Ungerade Zahlen Problem.txt“ vorgeschlagen werden. Diese Funktionen sollen es den Studierenden einfacher machen ihre Arbeit zu pausieren und wieder aufzunehmen. Im Vergleich zum Mockup (Abb. 17) wurden die Buttons zum Speichern und Laden näher an den Editor gerückt, damit der direkte Bezug dazu klarer ist. Zusätzlich wurden die Buttons unterschiedlich eingefärbt, um sie besser unterscheiden zu können und damit zu vermeiden, dass aus Versehen der falsche Button geklickt wird.

Mit dem „Abschicken“-Button (rote Markierung Nummer 6, Abb. 20) kann der Reduktionscode schlussendlich überprüft werden. Sobald dieser Button geklickt wurde, wird der Reduktionscode an den Reduktionsverifizierer gesendet. Dieser schickt dann eine Rückmeldung, von der die relevanten Informationen für die Studierenden angezeigt werden. Steht nur der Standardcode im Editor, also nur ein Kommentar, wie in Abbildung 19, dann gibt der Reduktionsverifizierer zurück, dass keine Reduktionsfunktion angegeben wurde. Den Studierenden wird dies wie in Abbildung 23 mitgeteilt. Was vielleicht aufgefallen ist, die Rückmeldung ist auf Englisch. Die relevanten Informationen der Rückmeldungen des Reduktionsverifizierers werden im Frontend nicht verändert. Der Reduktionsverifizierer und alle seine Ausgaben, wurden zunächst auf Englisch erstellt. Für die Zukunft stehen die Chancen gut, dass der Reduktionsverifizierer noch weiter verbessert wird und es dann auch Rückmeldungen in deutscher Sprache gibt.

Gibt es einen Syntaxfehler im Code wird dieser wie in Abbildung 24, mit Zeilennummer angezeigt. Wurde vom Reduktionsverifizierer kein Fehler im Reduktionscode gefunden sieht die Rückmeldung wie in Abbildung 25 aus. Da der Reduktionsverifizierer nicht verlässlich bestimmen kann, ob eine Re-

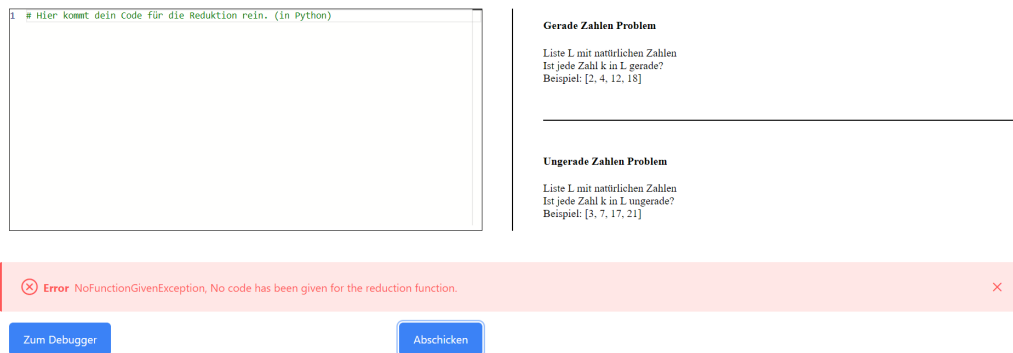


Abbildung 23: Screenshot Rückmeldung

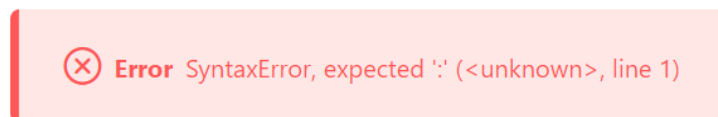


Abbildung 24: Screenshot Rückmeldung Syntaxfehler

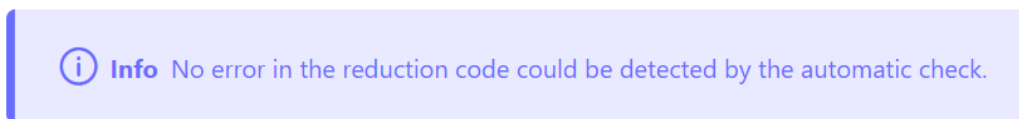


Abbildung 25: Screenshot Rückmeldung keinen Fehler gefunden

duktion korrekt ist, wird im Frontend nur eine Info angezeigt, welche aussagt, dass kein Fehler gefunden wurde, statt anzuzeigen, dass die Reduktion tatsächlich korrekt ist.

Mit dem „Zum Debugger“-Button (rote Markierung Nummer 7, Abb. 20) wird die Ansicht, wie der Name schon sagt, zum Debugger gewechselt.

Wie in Kapitel 3.2 erläutert, ist der Debugger dafür da, um Reduktionscode auf beliebige Eingaben zu testen. Im Vergleich zum Mockup (Abb. 26) wurden in der Umsetzung (Abb. 27), abgesehen von der Überschrift und der Beschreibung, die sich auf allen Ansichten geändert haben, die Buttons zum Speichern und Laden verschoben. Die neue Position dieser Buttons wurde vor allem so bestimmt, da sie so einheitlich zum Trainer (Abb. 19) in der oberen linken Ecke der Ansicht positioniert sind. Ansonsten unterscheidet sich die Umsetzung vom Mockup noch in der leichten Änderung der Pfeile in der Mitte der Ansicht und dadurch, dass jedes Feld im Mockup gefüllt ist. Letzteres ist in Abbildung 27 nicht der Fall, denn da soll erstmal die Stan-

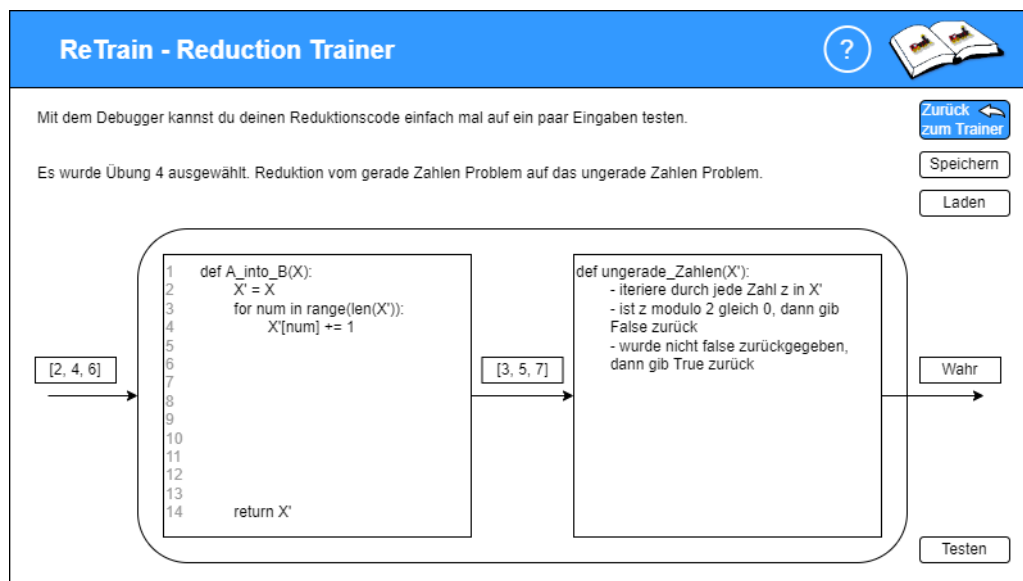


Abbildung 26: Mockup Debugger

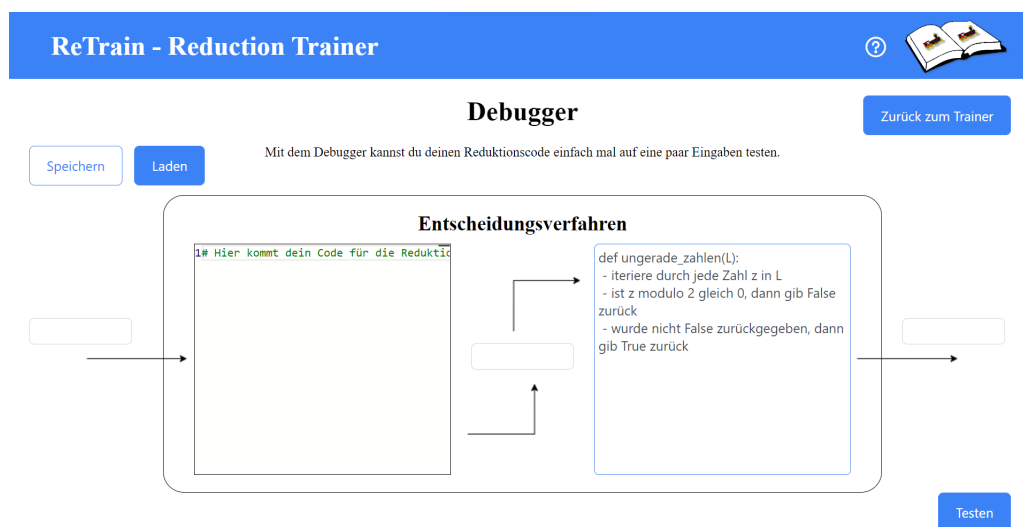


Abbildung 27: Screenshot Debugger

dardansicht gezeigt werden. Was auffällt ist, dass der Code im Editor (rote Markierung Nummer 3, Abb. 28) derselbe ist wie schon im Trainer (rote Markierung Nummer 4, Abb. 20). Beim Wechsel zwischen Trainer und Debugger wird der Code jedes mal übernommen, sodass auch ohne zwischendurch zu speichern beliebig hin und her gewechselt werden kann.

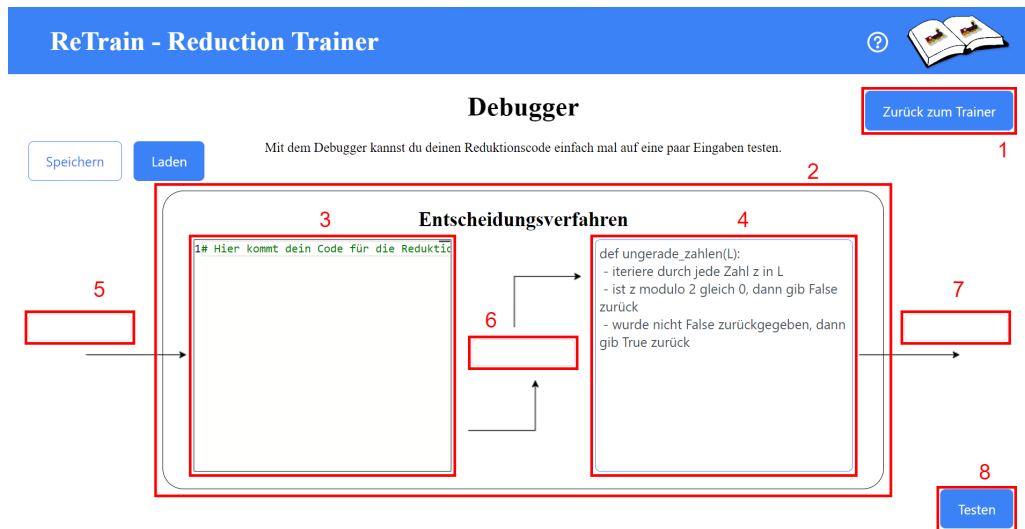


Abbildung 28: Screenshot Debugger mit Markierungen

„Zurück zum Trainer“ gelangen die Studierenden mit dem gleichnamigen Button (rote Markierung Nummer 3, Abb. 28) der sich am oberen rechten Rand der Ansicht befindet.

Schon im Kapitel 3.2 wurde erläutert welche Idee hinter dem Design des Debuggers steckt und genauso wurde es dann auch umgesetzt. Das große Rechteck mit den abgerundeten Ecken (rote Markierung Nummer 2, Abb. 28) in der Mitte der Ansicht stellt das neue Entscheidungsverfahren dar, welches mit einer korrekten Reduktion erreicht werden soll. Dieses besteht aus der Reduktionsfunktion (rote Markierung Nummer 3, Abb. 28) und dem Entscheidungsverfahren (rote Markierung Nummer 4, Abb. 28) des Problems, auf das reduziert wird. Das Entscheidungsverfahren welches mit Hilfe der Reduktion geschaffen werden soll (rote Markierung Nummer 2, Abb. 28), wird im Folgenden das neue Entscheidungsverfahren genannt und das Entscheidungsverfahren auf das reduziert wird, wird im Folgenden das gegebene Entscheidungsverfahren (rote Markierung Nummer 4, Abb. 28) genannt.

Für das gegebene Entscheidungsverfahren wird den Studierenden nur ein Pseudoalgorithmus angegeben. Dieser wurde von den Dozenten angelegt. Das dient vor allem zu einfacheren Lesbarkeit. Zusätzlich hat es aber auch den Effekt, dass die Dozenten mehr Freiheit bei der Erstellung des Entscheidungsverfahrens haben, da der Code nicht eingesehen werden kann.

Die kleinen rechteckigen Kästchen (rote Markierung Nummer 5, 6 und 7, Abb. 28) vor, im und nach dem neuen Entscheidungsverfahren sind dazu da, um einzelne Eingaben auszuwerten. Von links nach rechts das erste Kästchen (rote Markierung Nummer 5, Abb. 28) muss von den Studierenden mit einer Eingabe gefüllt werden. Die Eingabe in das Entscheidungsverfahren ist auch die Eingabe in die Reduktionsfunktion. Im zweiten Kästchen (rote Markierung Nummer 6, Abb. 28) steht die Ausgabe der Reduktionsfunktion, was auch gleichzeitig die Eingabe in das gegebene Entscheidungsverfahren (rote Markierung Nummer 4, Abb. 28) ist. Im dritten Kästchen (rote Markierung Nummer 7, Abb. 28) steht die Ausgabe des gegebenen Entscheidungsverfahrens, welches auch gleichzeitig die Ausgabe des neuen Entscheidungsverfahrens ist. In dem Kästchen steht also „true“, wenn die Eingabe in das gegebene Entscheidungsverfahren, also das Kästchen in der Mitte, eine Instanz davon ist. Gleichzeitig ist genau dann wenn die Reduktionsfunktion korrekt ist, auch die Eingabe in das neue Entscheidungsverfahren, also das Kästchen ganz links, eine Instanz des neuen Entscheidungsverfahrens, wenn im Kästchen ganz rechts „true“ steht. Genauso sind es jeweils keine Instanzen der Entscheidungsverfahren, wenn im Kästchen ganz rechts „false“ steht. Leichter ist das Ganze an einem Beispiel zu verstehen, welches nun folgt.

Um den Debugger zu benutzen, muss von den Studierenden zuerst eine Reduktionsfunktion im Editor (rote Markierung Nummer 3, Abb. 28) angegeben werden. Anschließend muss eine korrekte Eingabe in das kleine rechteckige Kästchen (rote Markierung Nummer 5, Abb. 28) ganz links geschrieben werden. Danach kann auf den „Testen“-Button geklickt werden. Die Eingabe, der Reduktionscode, und der Name des gegebenen Entscheidungsverfahrens werden nun zum Reduktionsverifizierer gesendet. Mit diesen Informationen führt der Reduktionsverifizierer den Reduktionscode und das gegebene Entscheidungsverfahren aus und gibt die Ergebnisse zurück. Im Frontend wird nun die Ausgabe des Reduktionscodes, also das kleine Kästchen in der Mitte der Ansicht (rote Markierung Nummer 6, Abb. 28) und die Ausgabe des gegebenen und somit auch des neuen Entscheidungsverfahrens, also das kleine Kästchen ganz rechts (rote Markierung Nummer 7, Abb. 28) gefüllt. Für die Reduktion vom „Gerade Zahlen Problem“ auf das „Ungerade Zahlen Problem“ sieht das ganze dann wie in Abbildung 29 aus. Wie die Eingabe in das neue Entscheidungsverfahren, welche von den Studierenden getätigt werden muss, aussehen soll ist nicht offensichtlich. Schon in der Traineransicht wurde erläutert, dass die Entscheidungsverfahren so codiert sind, dass sie nur ein einziges Tupel als Parameter entgegen nehmen. Das allein macht es aber noch nicht einfach zu wissen wie genau die Eingabe aussehen soll. Deshalb gibt es dafür etwas Unterstützung vom Frontend. Wird auf das Kästchen ganz

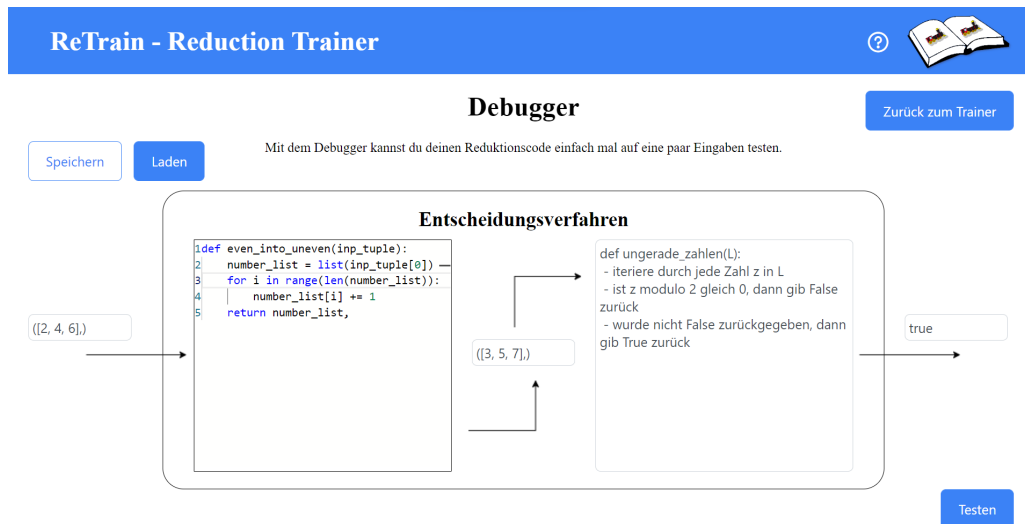


Abbildung 29: Screenshot Debugger mit Eingabe und Reduktionsfunktion

links (rote Markierung Nummer 5, Abb. 28) geklickt, öffnet sich ein kleines Fenster, welches in Abbildung 30 zu sehen ist. In Abbildung 30 wurde schon



Abbildung 30: Screenshot Debugger Eingabedialog

die Eingabe für das vorherige Beispiel (Abb. 29) getätigt. Wie die Eingabe aussehen soll steht direkt über dem Eingabefeld und wurde so von den Dozenten angegeben. In diesem Beispiel ist das *tuple[list[int]]*. Zusätzlich gibt es auch noch zwei Tooltips, einen für das bessere Verständnis der Notation in der Programmiersprache Python (Abb. 31) und einen zweiten, welcher ein Beispiel für eine Eingabe angibt (Abb. 32).

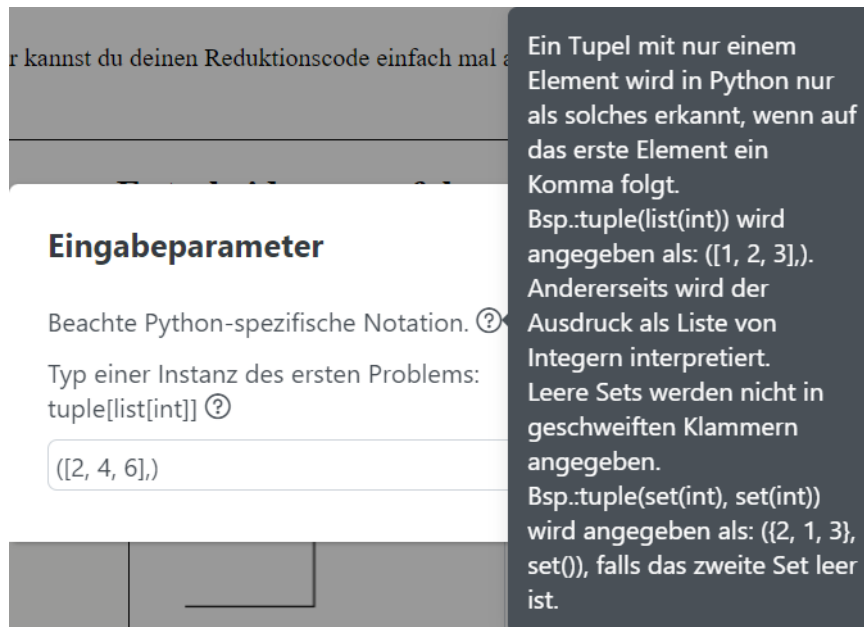


Abbildung 31: Screenshot Debugger Eingabedialog erster Tooltip

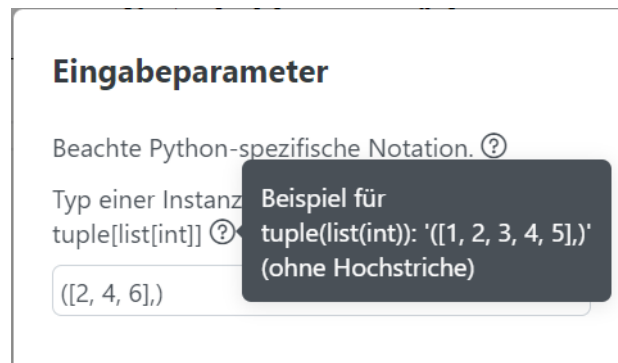


Abbildung 32: Screenshot Debugger Eingabedialog zweiter Tooltip

Falls beim Testen (rote Markierung Nummer 8, Abb. 28), also ausführen des Debuggers, vom Reduktionsverifizierer ein Fehler in der Eingabe oder im Reduktionscode gefunden wurde, wird er wie in Abbildung 33 angezeigt. Die Fehlermeldung dieser Abbildung weist darauf hin, dass die Eingabe, welche `[2, 4, 6]` ist, kein Tupel sondern eine Liste ist. Wie auch schon im Tooltip in Abbildung 31 steht, muss in einem Tupel mit nur einem Element auf das erste Element ein Komma folgen. Richtig wäre also, wie auch schon zuvor gesehen, die Eingabe `([2, 4, 6],)`. Im Gegensatz zum Trainer folgt beim Debugger keine zusätzliche Rückmeldung, dass kein Fehler gefunden wurde. Dass kein Fehler

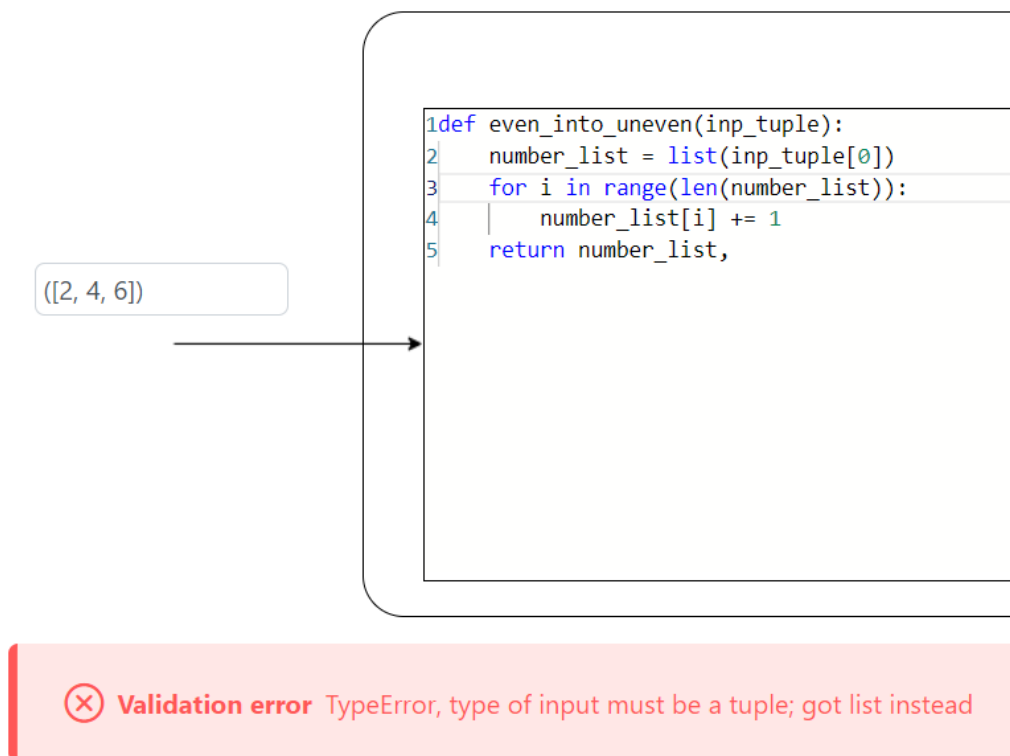


Abbildung 33: Screenshot Debugger Fehlermeldung, fehlerhafte Eingabe

gefunden wurde, kann daran festgestellt werden, dass die kleinen Kästchen in der Mitte und ganz rechts (rote Markierung Nummer 6 und 7, Abb. 28) aktualisiert wurden. Im nächsten Kapitel wird dann die Benutzeroberfläche der Dozenten behandelt, in der sich einige Designentscheidungen wiederfinden lassen, welche in diesem Kapitel schon erläutert wurden.

4.2 Benutzeroberfläche der Dozenten

Dieses Kapitel befasst sich mit dem Teil, der nur für die Dozenten sichtbar ist. Auch hier wurde schon zuvor im Kapitel 3.3 besprochen wie diese Benutzeroberfläche aufgebaut ist (Abbildung 5) und welche Funktionen für sie vorgesehen sind.

Das Erste, was die Dozenten sehen, ist die Ansicht, um sich anzumelden. Auch diese Ansicht hat sich beim Vergleich von Mockup (Abb. 34) und Umsetzung (Abb. 35) kaum verändert. Wie auf allen Ansichten wurde eine Unterüberschrift hinzugefügt und die Überschrift wurde in „Anmeldung“ geändert, um besser zum anderen, deutschen Text zu passen. Statt dem Wort „Passwort“

The mockup shows a blue header bar with the text "ReTrain - Reduction Trainer" on the left and an icon of an open book on the right. Below the header, the word "Login" is centered. Underneath, the label "Passwort:" is followed by a white rectangular input field. Below the input field is a blue button with the text "Anmelden".

Abbildung 34: Mockup Anmeldung

The screenshot shows the same blue header bar with "ReTrain - Reduction Trainer" and the book icon. Below the header, the word "Anmeldung" is centered. Underneath, the text "Melde dich zunächst an." is displayed. Below this is a white input field with the placeholder text "Passwort" and a small eye icon on the right side. Below the input field is a blue button with the text "Anmelden".

Abbildung 35: Screenshot Anmeldung

über dem Passwortfeld steht das Wort nur noch im Feld und das Passwort lässt mit dem Auge am Rand des Passwortfelds sichtbar oder wieder unsichtbar machen. Wie schon im Kapitel 4.1 erläutert, sind die Funktionen in der Leiste am oberen Rand auch im Frontend für die Dozenten die Gleichen, mit

Ausnahme, dass die Ansicht beim Klick auf die Leiste wieder zur Anmeldung gewechselt wird.

Die Anmeldung funktioniert wie folgt. Wird ein Passwort eingegeben und dann der „Anmelden“-Button (Abb. 35) geklickt, wird das Passwort an das Backend, welches auch die Datenbank verwaltet, gesendet. Dort wird es dann mit dem dort liegenden gehashten Passwort verglichen. Dass ein Passwort „gehasht“ wurde, heißt nur, dass es so verändert wurde, dass nicht erkannt werden kann was das eigentliche Passwort ist. Wurde festgestellt, dass das richtige Passwort im Frontend eingegeben wurde, wird der/die Dozent/in zur nächsten Seite weitergeleitet. War das Passwort fehlerhaft, zum Beispiel wenn einfach nichts in das Passwortfeld eingetragen wurde, dann wird eine Fehlermeldung wie in Abbildung 36 ausgegeben. Wird versucht auf irgendeine

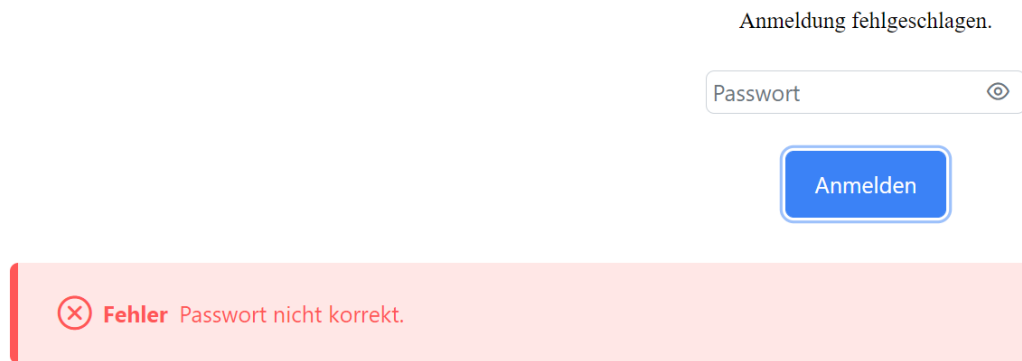



Abbildung 36: Screenshot Anmeldung Passwort inkorrekt

Seite zuzugreifen, welche erst nach der Anmeldung zugänglich ist, wird diese Person wieder zur Anmeldeseite umgeleitet. Wird das Passwort korrekt eingegeben, wird der/die Dozent/in zur Ansicht, in der die Übungen verwaltet werden können, weitergeleitet.

Wie vielleicht bemerkt wurde sieht die „Übungen verwalten“-Ansicht (Abb. 38) im Dozenten-Frontend der „Übungsauswahl“-Ansicht (Abb. 8) im Studierenden-Frontend ziemlich ähnlich. Hier werden nämlich wieder die Übungen beim Öffnen der Ansicht aus der Datenbank geholt, indem mit dem Backend kommuniziert wird, welches die Datenbank verwaltet. Für jede Übung wird den Dozenten auch noch eine Id angezeigt, welche für die Studierenden nicht zu sehen ist. Das sorgt dafür, dass Dozenten auch mehrere Übungen mit dem gleichen Namen anlegen können, denn nur die Id einer Übung muss einzigartig sein. Im zugehörigen Mockup (Abb. 37) wird die Id der Übung den Dozenten nicht angezeigt.

ReTrain - Reduction Trainer


Hier kannst du die Übungen verwalten, die die Studierenden sehen sollen.

Entscheidungsprobleme bearbeiten



Anzeigen

Übung 1	Löschen	CNF-SAT	\leq_p	3CNF-SAT	Bearbeiten	<input checked="" type="checkbox"/>
Übung 2	Löschen	3CNF-SAT	\leq_p	Independent Set Problem	Bearbeiten	<input checked="" type="checkbox"/>
Übung 3	Löschen	Independent Set Problem	\leq_p	Vertex Cover Problem	Bearbeiten	<input type="checkbox"/>

+

Abbildung 37: Mockup Übungen verwalten

ReTrain - Reduction Trainer

Übungen verwalten

Entscheidungsprobleme bearbeiten

Hier kannst du die Übungen verwalten, die die Studierenden sehen sollen.

Id der Übung	Name der Übung		Name Problem 1	Typ der Reduktion	Name Problem 2		Anzeigen
even_uneven	Übung 4	Löschen	Gerade Zahlen Problem	polynomialzeit	Ungerade Zahlen Problem	Bearbeiten	<input checked="" type="checkbox"/>
CNF-SAT_auf_3C... SAT	Übung 1	Löschen	CNF-SAT	polynomialzeit	3CNF-SAT	Bearbeiten	<input checked="" type="checkbox"/>
3CNF-SAT_auf_Ind...	Übung 2	Löschen	3CNF-SAT	polynomialzeit	Independent Set Problem	Bearbeiten	<input checked="" type="checkbox"/>
Independent...	Übung 3	Löschen	Independent Set Problem	polynomialzeit	Vertex Cover Problem	Bearbeiten	<input checked="" type="checkbox"/>

+

Abbildung 38: Screenshot Übungen verwalten

Die Checkbox am Ende jeder Reihe der Tabelle (rote Markierung Nummer 1, Abb. 39) befähigt die Dozenten dazu, die Übung in der Reihe für die Studierenden anzeigen zu lassen. In der Abbildung 38 beziehungsweise 39 sind alle Checkboxes ausgewählt, weshalb für die Studierenden in der Übungsauswahl (Abb. 8) auch all diese Übungen angezeigt werden. Nach der Erstellung einer

ReTrain - Reduction Trainer

?

Übungen verwalten

Entscheidungsprobleme bearbeiten

Hier kannst du die Übungen verwalten, die die Studierenden sehen sollen.

Id der Übung	Name der Übung		Name Problem 1	Typ der Reduktion	Name Problem 2		Anzeigen
even_uneven	Übung 4	Löschen	Gerade Zahlen Problem	polynomialzeit	Ungerade Zahlen Problem	Bearbeiten	<input checked="" type="checkbox"/>
CNF-SAT_auf_3C... SAT	Übung 1	Löschen	CNF-SAT	polynomialzeit	3CNF-SAT	Bearbeiten	<input checked="" type="checkbox"/>
3CNF-SAT_auf_Ind...	Übung 2	Löschen	3CNF-SAT	polynomialzeit	Independent Set Problem	Bearbeiten	<input checked="" type="checkbox"/>
Independent...	Übung 3	Löschen	Independent Set Problem	polynomialzeit	Vertex Cover Problem	Bearbeiten	<input checked="" type="checkbox"/>
+							

Abbildung 39: Screenshot Übungen verwalten mit Markierungen

Übung ist die Checkbox der Übung standardmäßig leer, damit nicht unbeabsichtigt Übungen angezeigt werden, die noch nicht angezeigt werden sollen. Es gibt auch den Fall, dass Übungen noch unvollständig sind. Übungen können auch unvollständig von Dozenten abgespeichert werden, also so, dass noch wichtige Details fehlen, die die Studierenden zum Lösen der Übung unbedingt brauchen. In dem Fall sollen die Übungen den Studierenden allerdings nicht angezeigt werden. Die Checkbox ist dann deaktiviert, sie lässt sich also nicht auswählen und wenn der Mauszeiger über sie gehalten wird, wird ein Hinweis in Form eines Tooltips angezeigt (Abb. 40). Mit dem „Löschen“-Button

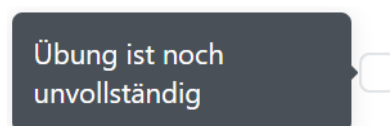


Abbildung 40: Screenshot Tooltip bei unvollständiger Übung

(rote Markierung Nummer 2, Abb. 39) kann, wie der Name schon sagt, die Übung in der Reihe des Buttons aus der Datenbank gelöscht werden. Wird auf den „Löschen“-Button in der obersten Reihe geklickt, öffnet sich zunächst ein Dialogfenster wie in Abbildung 41. 40). In diesem Fenster soll die Entscheidung diese Übung zu löschen bestätigt werden, weshalb auch nochmal Id und Name der zugehörigen Übung angezeigt werden. Mit einem Klick auf

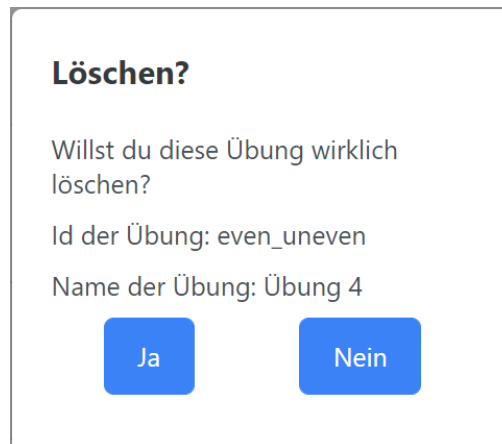


Abbildung 41: Screenshot Löschen Dialog

den „Nein“-Button wird das Fenster einfach wieder ohne weitere Aktion geschlossen. Wird stattdessen auf den „Ja“-Button geklickt, wird dem Backend die Id der Übung geschickt, die gelöscht werden soll. Das Backend löscht dann die Übung aus der Datenbank und schickt eine Bestätigung zurück. Im Frontend wird das Fenster wieder geschlossen und bei erfolgreichem Löschvorgang wird die Übung direkt aus der Tabelle entfernt. Mit einem Klick auf den großen „+“-Button direkt unter der letzten Übung wechseln die Dozenten zur „Übung erstellen“-Ansicht.

In der „Übung erstellen“-Ansicht kann von Dozenten eine neue Übung angelegt werden. Die Ansicht wird aufgrund ihrer Größe in mehreren Teilen beschrieben. Der Teil, der zuerst zu sehen ist, wurde wie in Abbildung 42 geplant und wie in Abbildung 43 umgesetzt. Da in dieser Ansicht viele Informationen eingegeben werden können, wird beim Versuch, die Browser Seite neu zu laden, ein Dialogfenster geöffnet. In diesem Dialogfenster wird gefragt, ob die Seite wirklich neu geladen werden soll. Die Dozenten haben dann noch die Möglichkeit, den Vorgang abubrechen und die eingegebenen Informationen zu speichern. Diese Vorsichtsmaßnahme gibt es auch auf allen anderen Ansichten im Dozenten Frontend, in denen Informationen eingegeben werden können.

Am oberen rechten Rand der Ansicht ist wieder ein Button zur Navigation. Dieser Button, der „Abbrechen“-Button, bringt die Dozenten beim Anklicken zurück zur „Übungen verwalten“-Ansicht. Wie in den Abbildungen zu sehen ist, ist die „Übung erstellen“-Ansicht wie ein Formular aufgebaut, in dem Zeile für Zeile die Eingaben eingetragen werden können. Dabei steht links die

ReTrain - Reduction Trainer

Erstelle eine neue Übung:

Abbrechen

Daten einer anderen Übung laden?

Keine Übung ausgewählt

(Vorsicht! Vorhandene Einträge werden überschrieben.)

Id der Übung

Name der Übung

Typ der Reduktion

ohne Restriktion

Problem 1:

Kein Problem ausgewählt

(Vorsicht! Vorhandene Einträge werden überschrieben.)

Name

Beschreibung

Gegeben

Frage

Beispiel

Entscheidungsverfahren (Python)

1

Abbildung 42: Mockup Übung erstellen

ReTrain - Reduction Trainer

Übung erstellen

Abbrechen

Erstelle eine neue Übung.

Daten einer anderen Übung laden?

Keine Übung geladen

(Vorsicht! Vorhandene Einträge werden überschrieben.)

Id der Übung

Id der Übung

Name der Übung

Name der Übung

Typ der Reduktion

Reduktion ist nicht in polynomialzeit

Problem 1:

Kein Problem wurde ausgewählt

(Vorsicht! Vorhandene Einträge werden überschrieben.)

Name

Name

Beschreibung

Beschreibung

Pseudocode für Debugger

Pseudocode für Debugger

Typ der Probleminstanz

1

Abbildung 43: Screenshot Übung erstellen

Bezeichnung für das, was im jeweiligen Feld rechts daneben eingetragen werden soll. Zusätzlich steht in den Eingabefeldern Text als Platzhalter, welcher aussagt, was in das Feld hineinkommt. Die Funktionen der Ansicht werden nun zeilenweise erläutert.

In der ersten Zeile mit der Bezeichnung „Daten einer anderen Übung laden?“ folgt ein Drop-down-Menü (Abb. 44), in dem eine bereits bestehende Übung ausgewählt werden kann. Dafür wird mit dem Backend kommuniziert und alle

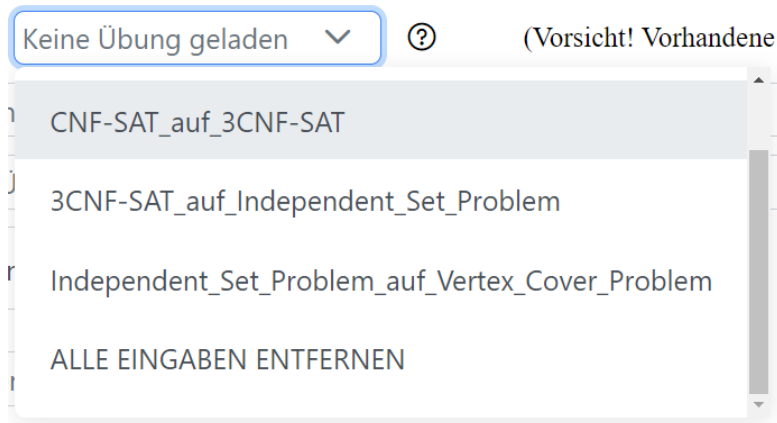


Abbildung 44: Screenshot Übung laden Drop-down-Menü

Übungen werden aus der Datenbank angefordert. Von diesen werden dann die Namen im Drop-down-Menü angezeigt. Wird eine Übung ausgewählt, werden alle nachfolgenden Eingaben, bis auf „Id der Übung“ und „Name der Übung“, mit den Informationen der ausgewählten Übung überschrieben. Diese Funktion kann zum Beispiel genutzt werden, wenn eine Übung nur leicht abgeändert werden soll oder damit bestimmte Eingaben nicht erneut eingegeben werden müssen. Wie schon als Hinweis in der gleichen Zeile steht, ist Vorsicht geboten, denn bestehende Eingaben werden überschrieben. Als letztes Element des Drop-down-Menüs steht immer „ALLE EINGABEN ENTFERNEN“ womit alle Eingabefelder wieder geleert werden können, wobei auch hier „Id der Übung“ und „Name der Übung“ unberührt bleiben. Die beiden unberührten Eingabefelder werden nicht überschrieben, da es nicht sinnvoll ist zwei identische Übungen in der Datenbank zu speichern. Diese Felder sollen also unbedingt neu ausgefüllt werden. Wird der Mauszeiger über das Fragezeichensymbol direkt nach dem Menü gehalten, so erscheint ein Tooltip (Abb. 45), der genau darauf hinweist. Das Fragezeichensymbol und der Tooltip wurden im Mockup noch nicht vorgesehen.

In den nachfolgenden beiden Zeilen soll nun die Id und der Name der Übung in die Eingabefelder geschrieben werden. Wie schon zuvor erwähnt, muss die Id jeder Übung einzigartig sein, um sie damit unterscheiden zu können. Der Name der Übung der hier eingetragen wird, wird dann den Studierenden angezeigt.

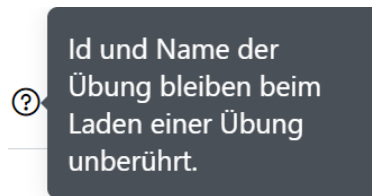


Abbildung 45: Screenshot Übung laden Tooltip

In der nachfolgenden Zeile, in der „Typ der Reduktion“ steht, kann mit einem Klick auf den Button verändert werden, ob die Reduktion in Polynomialzeit sein soll, wie in Abbildung 46, oder nicht, wie in Abbildung 43. Im Mockup

Typ der Reduktion

Reduktion ist in polynomialzeit

Abbildung 46: Screenshot Typ der Reduktion ändern Button

(Abb. 42) war an dieser Stelle noch ein Drop-down-Menü, um die Möglichkeit offen zulassen, die Reduktion in Zukunft noch anders zu beschränken, zum Beispiel in logarithmischen Platz. Da der Einsatz des Reduktions-Trainers allerdings vor allem in der grundlegenden Veranstaltung „Berechenbarkeit und Komplexität“ geplant ist, reicht es wahrscheinlich zum derzeitigen Stand eine Reduktion in Polynomialzeit beschränken zu können. Um zwischen „in Polynomialzeit“ und „nicht in Polynomialzeit“ wechseln zu können, ist ein Button dann praktischer als ein Drop-down-Menü.

Als Nächstes sollen Angaben zum ersten Problem der neuen Übung gemacht werden. Wie schon für die neue Übung selbst, gibt es auch wieder ein Drop-down-Menü, womit diesmal bestehende Probleme in die kommenden Eingabefelder geladen werden können. Die Funktionen des Drop-down-Menüs sind fast dieselben wie schon beim Menü in der ersten Zeile der Ansicht, nur dass es diesmal Probleme statt Übungen sind und dass keine Eingabefelder unberührt bleiben und somit auch kein Tooltip (Tooltip aus erster Zeile Abb. 45) nötig ist.

In der ersten Zeile unter den Angaben für „Problem 1“ soll das Entscheidungsproblem einen Namen bekommen. Der Name ist für Probleme einzigartig, um sie zu unterscheiden.

Darauf folgt die Beschreibung. Was als Beschreibung von den Dozenten angegeben werden sollte, wurde schon im Studierenden-Frontend erläutert (rote

Markierung Nummer 3, Abb. 20). Kurz zusammengefasst soll das Entscheidungsproblem durch einen „Gegeben“- und einen „Frage“-Teil beschrieben werden. Allerdings kann das Beschreibungsfeld auch noch mit weiteren Informationen wie Beispielen des Entscheidungsproblems gefüllt werden. Geplant war zuerst, wie im Mockup in Abbildung 42 zu sehen, die Dozenten dazu zu bringen „Gegeben“, „Frage“, „Beschreibung“ und „Beispiel“ in extra Feldern anzugeben. Diese Felder wurden zu einem „Beschreibung“-Feld zusammengefasst, was den Dozenten die Freiheit gibt, die Information auch anders weitergeben zu können.

Nach der Beschreibung soll Pseudocode für den Debugger angegeben werden. Die Eingabefelder „Beschreibung“ und „Pseudocode für Debugger“ sind etwas größer als andere und skalieren im Gegensatz zu den schmaleren Feldern, bei vielen Zeilen Eingabe einfach vertikal mit. Wie schon im Debugger gesehen (Abb. 28) soll als Pseudocode hier ein Entscheidungsverfahren für dieses Problem angegeben werden. Damit ist für die Studierenden leicht zu verstehen wie das Entscheidungsverfahren funktioniert und der Code für das tatsächliche Entscheidungsverfahren muss nicht offengelegt werden. Für alle weiteren Eingaben des Problems stehen Code-Editoren zur Verfügung wie sie schon im Studierenden-Frontend beim Trainer und beim Debugger zu sehen waren.

Für den „Typ der Problemistanz“ steht nur ein schmaler Editor zur Verfügung, da der Typ wahrscheinlich nicht mehr als eine Zeile Platz braucht. Der „Typ der Problemistanz“ ist der Typ dessen was ein Entscheidungsverfahren als Eingabe entgegennimmt. Wie schon im Studierenden-Frontend beschrieben ist die Eingabe immer ein Tupel, welches aber verschiedene Datentypen als Elemente beinhalten kann. Für das Gerade Zahlen Problem, welches zuvor schon als Beispiel gedient hat, würde hier für den Typ `tuple[list[int]]` eingetragen werden. So wird es dann auch für die Studierenden im Debugger angezeigt, wie in Abbildung 30 zu sehen ist. Hinter dem Fragezeichensymbol in dieser Zeile verbirgt sich wie immer ein Tooltipp (Abb. 47). Wie in diesem

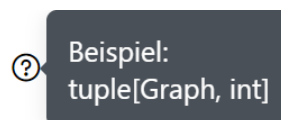


Abbildung 47: Screenshot Typ der Problemistanz Tooltipp

Tooltipp zu sehen ist können auch neue Datentypen eingetragen werden, die dann im nachfolgenden Feld „Definition der Datenklassen“ (Abb. 49) defi-

nirt werden. Der „Typ der Problemistanz“ ersetzt die Angabe einzelner

Abbildung 48: Mockup Übung erstellen (Teil 2)

Abbildung 49: Screenshot Übung erstellen (Teil 2)

Parameter, wie sie noch im Mockup (Abb. 48) geplant waren. Damit wurden die drei Zeilen „Name Parameter“, „Typ Parameter“ und „Parameter hinzufügen/entfernen“ auf eine Zeile reduziert, wobei für jeden Parameter zwei neue Zeilen hinzugekommen wären. Die Umsetzung ist also deutlich platzsparender und erzwingt, dass von den Studierenden die Parameter in der richtigen Reihenfolge angegeben werden müssen. Letzteres erleichtert die Verarbeitung im Reduktionsverifizierer.

Im Editor für die „Definition der Datenklassen“ können neue Klassen wie im Code Abschnitt 2 definiert werden. In der Datenklasse können dann beliebig viele Variablen angelegt werden, für die der Typ angegeben werden muss.

```

@dataclass
def <Name der Klasse>:
    <Name der Variable 1>: <Typ Variable 1>
    <Name der Variable 2>: <Typ Variable 2>
    ...

```

Code Abschnitt 2: Aufbau Definition der Datenklassen

Der Typ einer Variable kann auch wieder ein selbst erdachter Datentyp sein, welcher dann einfach auch nach diesem Schema als Datenklasse angelegt werden kann. Für den Datentyp *Graph*, welcher im vorherigen Tooltipp zu sehen war (Abb. 47), gibt es eine beispielhafte Definition im Tooltipp zur Beschriftung „Definition der Datenklassen“ (Abb. 50).

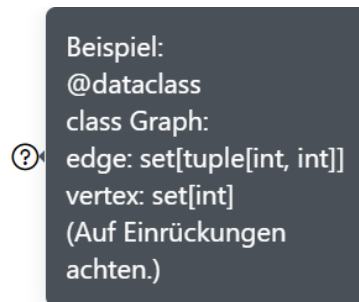


Abbildung 50: Screenshot Definition der Datenklassen Tooltipp

Nun kann im darunterliegenden Editor („Entscheidungsverfahren“, 49) noch ein Entscheidungsverfahren angegeben werden. Auch für das erste Problem, welches das Problem ist, von dem aus reduziert wird, muss ein Entscheidungsverfahren angegeben werden, damit der Reduktions-Trainer korrekt funktioniert. Obwohl das das Entscheidungsverfahren ist, das mit der Reduktion herbeigeführt werden soll, ist es dennoch wichtig auch dieses Entscheidungsverfahren anzugeben. Die Testfälle müssen nämlich zuerst mit einem korrekten Entscheidungsverfahren ausgewertet werden, um sie dann mit dem neuen Entscheidungsverfahren, welches die Studierenden durch die Reduktion erzeugen, vergleichen zu können. Zudem ist es hilfreich, das Entscheidungsverfahren in der Datenbank abzuspeichern, um später auch einfach auf dieses Problem reduzieren zu können. Ein Entscheidungsverfahren wird wie eine Funktionsdefinition in Python angegeben. Beachtet werden muss dabei nur, dass die Funktion ein einziges Tupel als Eingabeparameter bekommt und *True* oder *False* zurückgibt. Ein Beispiel dafür gibt es auch hier wieder im zugehörigen Tooltipp (Abb. 51).

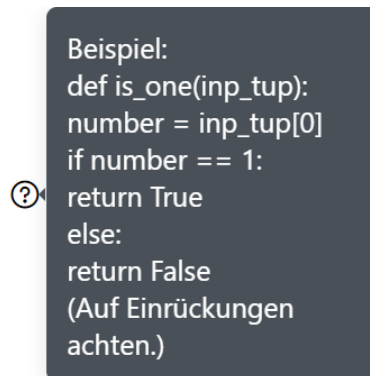


Abbildung 51: Screenshot Entscheidungsverfahren Tooltipp

Als Letztes können für das Problem noch Testfälle angegeben werden. Wie vorher schon angemerkt braucht es ein Entscheidungsverfahren, um dadurch die Testfälle laufen zu lassen. Für jeden Testfall kann dann gespeichert werden, ob er eine Ja- oder eine Nein-Instanz des Problems ist. Wenn der Trainer von den Studierenden genutzt wird, werden die Testfälle vom Reduktionsverifizierer in das neue, durch die Reduktion entstandene, Entscheidungsverfahren gegeben. Die Ergebnisse der Testfälle vom Entscheidungsverfahren, welches von den Dozenten angegeben wurde, können dann mit den Ergebnissen von dem durch die Reduktion entstandenen Entscheidungsverfahren verglichen werden. Sind die Ergebnisse nicht gleich, wird das den Studierenden mitgeteilt. Somit werden die Studierenden durch viele Testfälle und ein korrektes von den Dozenten angegebenes Entscheidungsverfahren, besser auf Fehler hingewiesen.

Wird eine Übung erstellt, dann sind natürlich keine Testfälle angegeben und es sieht aus wie in Abbildung 49. Mit dem „+“-Button kann ein neuer Testfall angelegt werden, wofür eine neue Zeile mit einem schmalen Editor-Fenster erscheint (Abb. 52). Auf diese Weise können beliebig viele Testfälle erstellt



Abbildung 52: Screenshot Testfälle hinzufügen oder entfernen

werden. In jeden Editor kann dann ein Testfall geschrieben werden. Mit einem Klick auf den „-“-Button wird die zuletzt hinzugefügte Zeile wieder entfernt. Ein Testfall soll nun so aussehen wie schon der zuvor angegebene „Typ der Problemistanz“, nur eben mit Werten statt Typen. Für das Gerade Zahlen

Problem, dessen Typ `tuple[list[int]]` ist, könnte ein Testfall `([1, 2, 3, 4, 5],)` sein. Dieser Testfall wäre eine Nein-Instanz des Gerade Zahlen Problems, denn in der Liste befinden sich nicht nur gerade Zahlen. Auch hier gibt es wieder einen Hinweis darauf, die Python-spezifische Notation zu beachten und wird der Mauszeiger über das Fragezeichensymbol gehalten, öffnet sich ein Tooltipp, welches in Abbildung 53 zu sehen ist.

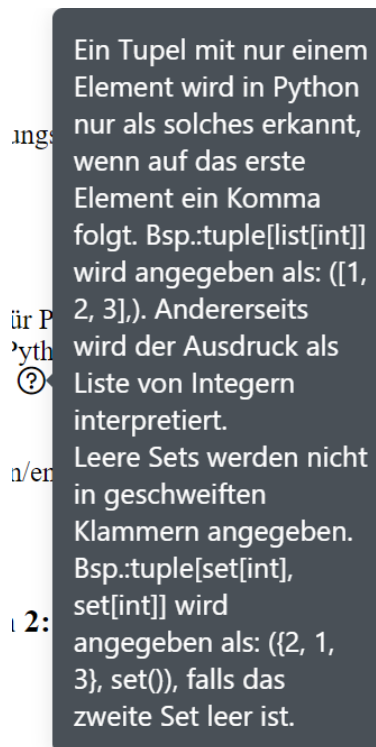


Abbildung 53: Screenshot Testfälle Tooltipp

Damit sind alle Eingaben des ersten Problems erläutert worden. Das Mockup in Abbildung 48 und der Screenshot in Abbildung 49 wurden so geschnitten, dass beide mit den Eingaben des ersten Problems enden. Das zweite Problem erfordert genau dieselben Eingaben wie das erste Problem, weshalb zu den Eingaben nichts weiter gesagt werden muss. Am unteren Ende der Ansicht, unter den Testfällen des zweiten Problems, befinden sich noch die Eingabe für Schwierigkeitsgrade und der „Übung erstellen“-Button.

Schwierigkeitsgrade sind, wie schon im Studierenden-Frontend angemerkt, Reduktionscode der von den Dozenten für das Bearbeiten der Reduktion vorgegeben wird. Genau wie bei den Testfällen sind zu Beginn noch keine

Schwierigkeitsgrade angelegt. Mit dem „+“-Button können neue Schwierigkeitsgrade hinzugefügt und mit dem „-“-Button die Neuesten wieder entfernt werden. Hier steht allerdings ein größerer Editor bereit, um mehrere Zeilen Code vorgeben zu können (Abb. 55). Wie im Mockup (Abb. 12) zu sehen

Mockup of the 'Übung erstellen' form. It includes a header 'Schwierigkeitsgrad hinzufügen/entfernen' with minus and plus buttons and a note '(Standardmäßig gibt es Schwierigkeitsgrad 0, Schwer, ohne gegebenen Code)'. Below are input fields for 'Name der Schwierigkeit' and 'Code der vorgegeben werden soll' (containing '1'). A blue 'Übung erstellen' button is at the bottom.

Abbildung 54: Mockup Übung erstellen (unterer Rand der Ansicht)

Screenshot of the 'Übung erstellen' form. It shows the 'Schwierigkeit hinzufügen/entfernen' header with plus and minus buttons. Below is a large text area for the difficulty level, with a small '1' in the top left corner. A blue 'Übung erstellen' button is at the bottom.

Abbildung 55: Screenshot Übung erstellen (unterer Rand der Ansicht)

ist, war geplant jeder Schwierigkeit eigenen Namen zu geben, weshalb auch ein Hinweis dazu neben den „+“- und „-“-Buttons steht. In der Umsetzung (Abb. 55) wurde dann dazu übergegangen die Schwierigkeitsgrade durchnummerieren, um den Dozenten einen Schritt zu ersparen.

Das letzte Element der Ansicht ist der „Übung erstellen“-Button. Wurden Eingaben für eine neue Übung mit neuen Problemen, welche alle noch nicht in der Datenbank liegen, getätigt, dann werden die Eingaben als Übung zusammengefasst und an das Backend gesendet. Dort wird die neue Übung und die Probleme nochmal extra in die Datenbank gelegt (genauer zur Datenbank im Kapitel 4.3). Das funktioniert allerdings nicht immer so einfach, denn bevor die Übung und deren Probleme in der Datenbank gespeichert werden, werden die Eingaben noch überprüft, was unter anderem der Datenkonsistenz dient.

Wird der „Übung erstellen“-Button geklickt, wird zunächst kontrolliert, ob alle wichtigen Eingaben getätigt wurden. Um eine Übung zu erstellen, müssen mindestens die Id der Übung und die Namen der Probleme angegeben werden. Wird dies nicht getan und der „Übung erstellen“-Button geklickt, resultiert das in einer Fehlermeldung, die über dem Button angezeigt wird. Zusätzlich werden die fehlenden Eingabefelder mit einem roten Rand versehen. Wird von den drei gerade erwähnten Eingaben also nur der Name für Problem 2 nicht eingegeben, wird die Fehlermeldung die in Abbildung 56 zu sehen ist ausgegeben und das Eingabefeld wie in Abbildung 57 rot umrandet. Auch wenn es in der Datenbank schon eine Übung mit dieser Id gibt, wird

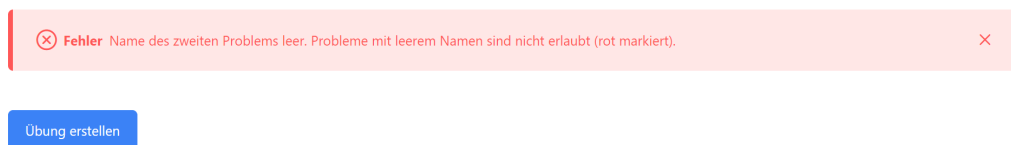


Abbildung 56: Screenshot Fehler Name des zweiten Problems fehlt

Problem 2:	Kein Problem wurde ausgewählt ▼	(Vorsicht! Vorhandene Einträge werden überschrieben.)
Name	<input type="text" value="Name"/>	
Beschreibung	<input type="text" value="Beschreibung"/>	

Abbildung 57: Screenshot fehlendes Eingabefeld rot umrandet

eine Fehlermeldung angezeigt. Dafür sendet das Backend beim Versuch, eine Übung mit der angegebenen Id in der Datenbank zu erstellen, einen Fehler zurück, welcher den Dozenten dann direkt angezeigt wird.

Die Id der Übung und die Namen der Probleme reichen zwar aus, um eine Übung zu erstellen, allerdings wird auch überprüft ob alle für die Studierenden nötigen Informationen vorhanden sind. Die nötigen Informationen schließen nur die Beschreibungen und Testfälle der Probleme sowie die Schwierigkeitsgrade der Übung aus. Alle anderen Informationen müssen angegeben werden, um die Übung überhaupt für die Studierenden anzeigen zu können. Wird die Übung in der Datenbank gespeichert, obwohl noch Informationen fehlen, wird in der „Übungen verwalten“-Ansicht die Checkbox zum Anzeigen der Übung deaktiviert, so wie es auch schon in Abbildung 40 zu sehen war. Fehlen Eingaben beim Klick auf den „Übung erstellen“-Button öffnet sich ein Dialogfenster (Abb. 58), um die Dozenten genau darauf hinzuweisen. Wird im Dialogfenster auf den „Abbrechen“-Button gedrückt, schließt sich

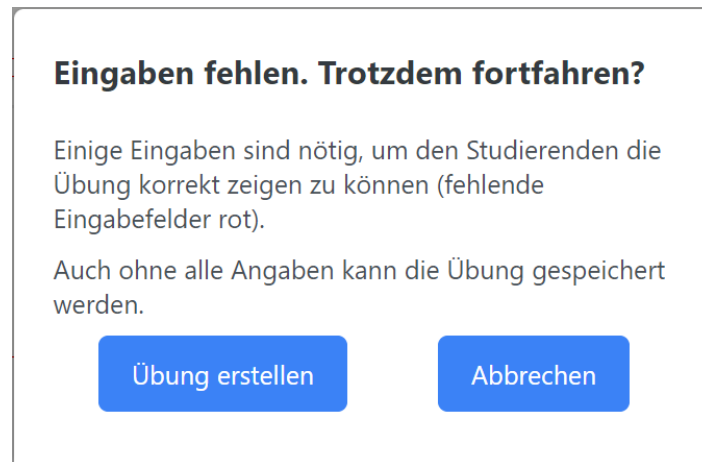


Abbildung 58: Screenshot fehlende Eingaben Dialogfenster

das Fenster wieder und fehlende Eingabefelder sind rot markiert. Beim Klick auf „Übung erstellen“ wird die Übung trotz fehlender Eingaben erstellt und die Ansicht wechselt zurück zur „Übungen verwalten“-Ansicht. Dort wird die neue Übung dann auch direkt angezeigt.

Das in Abbildung 58 zu sehende Dialogfenster ist aber nicht das einzige Dialogfenster, das sich beim Klick auf den „Übung erstellen“-Button öffnen kann. Falls zum Beispiel eine Übung geladen wurde, was mit dem Drop-down-Menü (Abb. 44) in der ersten Zeile der „Übung erstellen“-Ansicht möglich ist und danach Änderungen an einem Problem vorgenommen wurden, aber der Name des Problems gleich bleibt, dann öffnet sich ein Dialogfenster (Abb. 59), um zu überprüfen ob das Problem aktualisiert werden soll. Denn der

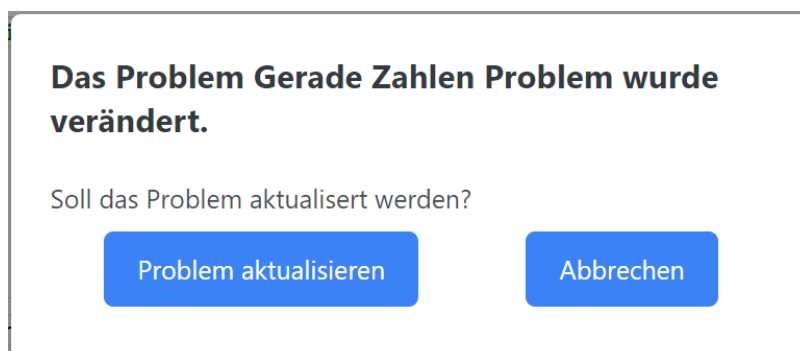


Abbildung 59: Screenshot Problem aktualisieren Dialogfenster

Name eines Problems ist einzigartig, was heißt, dass in der Datenbank keine

unterschiedlichen Probleme mit dem gleichen Namen liegen dürfen. Wird im Dialogfenster welches in Abbildung 59 zu sehen ist auf den „Problem aktualisieren“-Button gedrückt, dann wird das Problem mit dem gleichen Namen wie das angegebene Problem in der Datenbank überschrieben. Wird stattdessen auf „Abbrechen“ gedrückt, schließt sich das Dialogfenster einfach wieder. Falls beide Probleme verändert wurden, öffnet sich nachdem das eine Problem aktualisiert wurde, das Fenster erneut für das andere Problem. Um welches Problem es geht, ist in der Überschrift des Dialogs zu sehen.

Wurde schlussendlich eine neue Übung erstellt, wird zurück zur „Übungen verwalten“-Ansicht gewechselt. Mit dem „Bearbeiten“-Button (rote Markierung Nummer 4, Abb. 39), der für jede erstellte Übung zur Verfügung steht, kann zur „Übung bearbeiten“-Ansicht (Abb. 61) für die ausgewählte Übung gewechselt werden. Die „Übung bearbeiten“-Ansicht unterscheidet sich kaum von der „Übung erstellen“-Ansicht, weshalb nur auf die Unterschiede der beiden Ansichten eingegangen wird.

Wie im Mockup (Abb. 60) und im Screenshot (Abb. 61) der „Übung bearbeiten“-Ansicht zu sehen ist wurde die Übung 4 geladen. Dafür wurde wieder

ReTrain - Reduction Trainer

Du bearbeitest derzeit Übung 4. Abbrechen Löschen

Daten einer anderen Übung laden? Keine Übung ausgewählt (Vorsicht! Vorhandene Einträge werden überschrieben.)

Id der Übung:

Name der Übung:

Typ der Reduktion: polynomialzeit

Problem 1: Kein Problem ausgewählt (Vorsicht! Vorhandene Einträge werden überschrieben.)

Name:

Beschreibung:

Gegeben:



Frage:

Beispiel:

Entscheidungsverfahren (Python):

Abbildung 60: Mockup Übung bearbeiten

mit dem Backend kommuniziert, um sich die Übung aus der Datenbank zu holen und alle Eingabefelder wurden entsprechend der Übung aus der Datenbank gefüllt. Im Vergleich zur „Übung erstellen“-Ansicht hat sich geändert,

ReTrain - Reduction Trainer



Übung bearbeiten

Du bearbeitest derzeit die Übung (id): even_uneven

Abbrechen

Id der Übung

Name der Übung

Typ der Reduktion

Reduktion ist in polynomialzeit

Problem 1:

Name

Beschreibung

Liste L mit natürlichen Zahlen
Ist jede Zahl k in L gerade?
Beispiel: [2, 4, 12, 18]

Pseudocode für Debugger

```
def gerade_zahlen(L):
- iteriere durch jede Zahl z in L
- ist z modulo 2 ungleich 0, dann gib False zurück
- wurde nicht False zurückgegeben, dann gib True zurück
```

Typ der Probleminstanz

1

Abbildung 61: Screenshot Übung bearbeiten

dass in der Unterüberschrift die Id der Übung steht, welche gerade bearbeitet wird. Falls die Id der Übung geändert wird, ist somit immer noch die vorherige Id bekannt. Außerdem fehlt die Zeile, in der Übungen geladen werden können, genauso wie die Zeilen in der Probleme geladen werden können. Das dient für eine klare Abgrenzung der beiden Ansichten und sorgt dafür, dass in der „Übung bearbeiten“-Ansicht eher kleine Änderungen vorgenommen werden statt die Übung völlig neu zu schreiben. Im Mockup wurde daran noch nicht gedacht, denn da war noch geplant die Übung und Probleme auch in dieser Ansicht laden zu können. Auch der „Löschen“-Button, der oben rechts im Mockup (Abb. 60) zu sehen ist, wurde nicht umgesetzt, um zu vermeiden hier aus Versehen die Übung zu löschen und um den Ort, an dem Übungen gelöscht werden können, auf die „Übungen verwalten“-Ansicht zu beschränken. Das Einzige, was sich im Vergleich zur „Übung erstellen“ Ansicht noch verändert hat, ist, dass der Button ganz am unteren Ende der Ansicht nun mit „Änderungen bestätigen“ statt „Übung erstellen“ beschriftet ist. Die Funktion des Buttons hat sich an dieser Stelle teilweise verändert.

Mit dem „Änderungen bestätigen“-Button wird die gegebene Übung auch wieder auf fehlende Eingaben geprüft, so wie das schon beim Erstellen einer Übung war. Was sich ändert, ist, dass die geladenen Probleme bei Änderungen auch einfach so in der Datenbank überschrieben werden. Dabei wird nicht darauf hingewiesen, dass es dieses Problem schon in der Datenbank gibt, denn

davon ist beim Bearbeiten einer Übung auszugehen. Stattdessen öffnet sich ein Dialogfenster (Abb. 62), wenn sich der Name eines Problems ändert. In

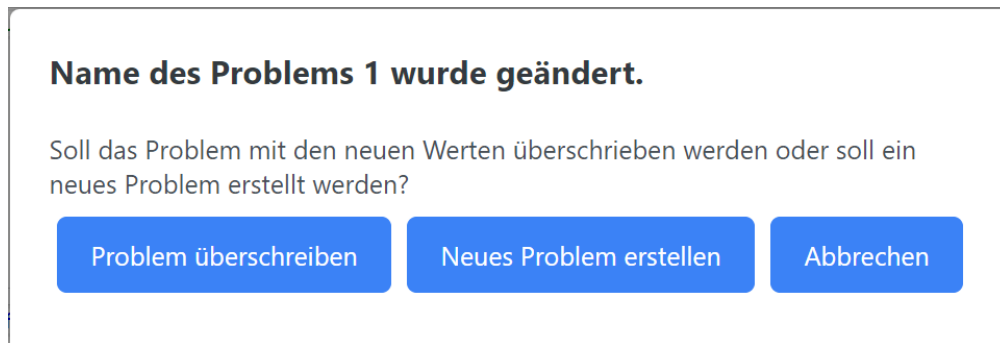


Abbildung 62: Screenshot Name eines Problems wurde verändert Dialogfenster

dem Fall wird gefragt, ob das Problem einen neuen Namen bekommen, also überschrieben werden soll oder ob ein neues Problem mit dem veränderten Namen erstellt werden soll. Der Vorgang kann natürlich auch abgebrochen werden. Sind alle Eingabefelder gefüllt und die Namen der Probleme wurden nicht verändert, werden mit dem „Änderungen bestätigen“-Button die Übung und deren Probleme in der Datenbank überschrieben. Egal ob neue Probleme erstellt wurden oder nicht, wird nach dem Bestätigen der Änderungen zurück zur „Übungen verwalten“-Ansicht gewechselt.

In der „Übungen verwalten“-Ansicht wurde nur ein Button noch nicht erwähnt und zwar der „Entscheidungsprobleme bearbeiten“-Button (rote Markierung Nummer 4, Abb. 39). Mit einem Klick auf diesen Button navigieren die Dozenten zur „Entscheidungsprobleme verwalten“-Ansicht (Abb. 64).

Diese Ansicht gibt es, um Probleme auch getrennt von Übungen verwalten zu können. Zum Beispiel wenn Dozenten einfach nur eine Idee für ein Problem haben, aber noch nicht direkt eine Übung damit erstellen wollen. Auch wenn es den Dozenten zu mühselig ist, direkt eine komplette Übung zu erstellen, ist es praktisch, Probleme getrennt anlegen zu können und diese später einfach in eine Übung zu laden. Beim Vergleich von Mockup (Abb. 63) und Umsetzung (Abb. 64) gibt es keine Veränderung, die nicht schon bei vorherigen Mockups angesprochen wurde. Die „Entscheidungsprobleme verwalten“-Ansicht ähnelt der „Übungen verwalten“-Ansicht, deren Funktionen und vor allem den Ansichten die von hier aus mit einem Klick erreicht werden können so sehr, dass hauptsächlich auf die wenigen Unterschiede eingegangen wird. Die Tabelle in der „Entscheidungsprobleme verwalten“-Ansicht, in welcher die

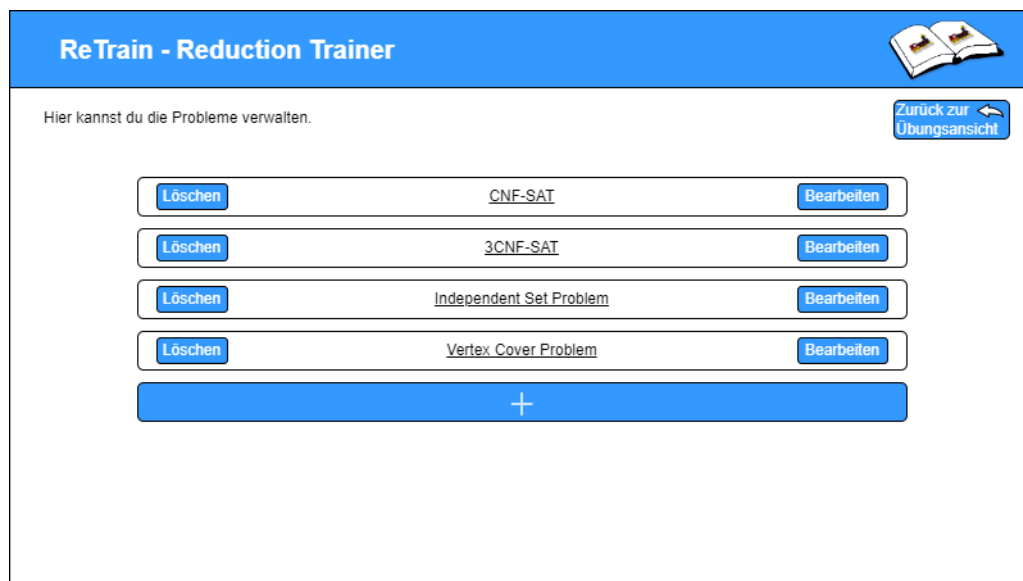


Abbildung 63: Mockup Entscheidungsprobleme verwalten

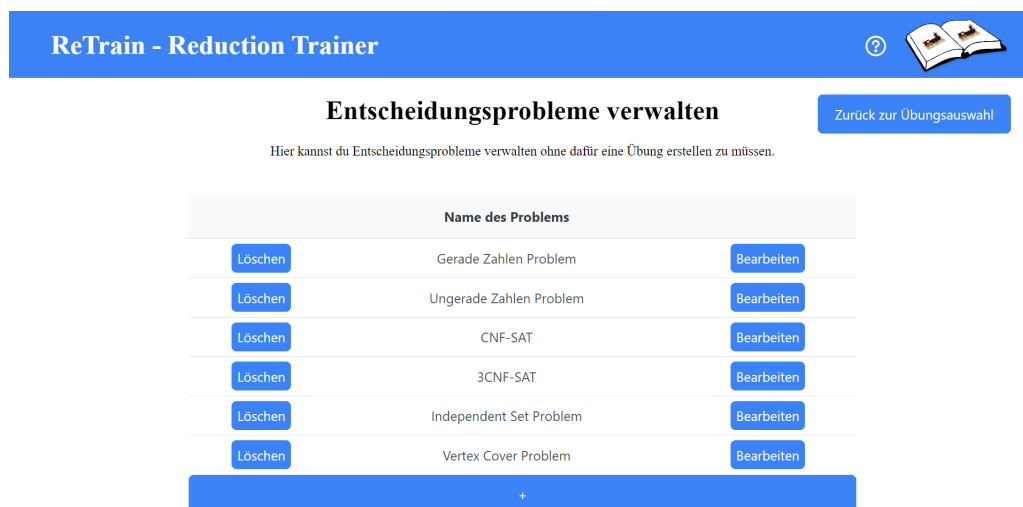


Abbildung 64: Screenshot Entscheidungsprobleme verwalten

Probleme aufgelistet werden, umfasst zu einem Problem dessen Namen sowie den „Löschen“- und den „Bearbeiten“-Button. Damit sind in der Tabelle deutlich weniger Informationen als bei den Übungen. Mit dem „Löschen“-Button wird wie auch bei den Übungen ein Dialogfenster geöffnet, welches nach Bestätigung der Aktion fragt. Besonders ist hier, dass wenn ein Pro-

blem gelöscht werden soll, also wenn im Dialogfenster bestätigt wurde, das Problem zu löschen, dann darf dieses Problem in keiner Übung vorhanden sein. Andernfalls erscheint eine Fehlermeldung über der Tabelle (Abb. 65). Das funktioniert wie folgt. Nachdem mit dem Backend kommuniziert wurde,

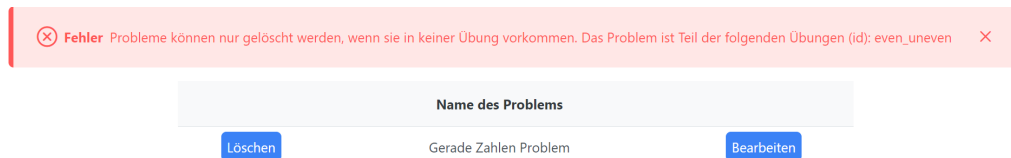


Abbildung 65: Screenshot Fehler Problem kommt in Übung vor

dass ein bestimmtes Problem gelöscht werden soll, wird im Backend überprüft ob das Problem Teil einer Übung ist. Ist das der Fall, wird ein Fehler zurückgegeben, der den Dozenten direkt angezeigt wird.

Mit dem großen „+“-Button geht es zur „Problem erstellen“-Ansicht, welche dem Teil in der „Übung erstellen“-Ansicht gleicht, in dem die Eingaben für ein Problem getätigt werden sollen. Soll ein Problem erstellt werden, dann wird im Unterschied zur Erstellung einer Übung nicht geprüft, ob alle Felder korrekt gefüllt sind, denn Probleme allein können den Studierenden nicht angezeigt werden, Übungen schon. Diese Kontrolle findet dann statt, wenn das Problem in eine Übung eingebunden wird. Was dennoch geprüft wird, ist, ob der Name eines Problems leer ist und ob es ein Problem mit dem angegebenen Namen schon in der Datenbank gibt. In beiden Fällen gibt das Backend einen Fehler zurück, der den Dozenten direkt angezeigt wird.

In der „Problem bearbeiten“-Ansicht wird dann das ausgewählte Problem geladen. Diese Ansicht unterscheidet sich von der „Problem erstellen“-Ansicht nur dadurch, dass kein Drop-down-Menü zum Laden eines anderen Problems angezeigt wird und dadurch, dass der „Problem erstellen“-Button nun mit „Änderungen bestätigen“ beschriftet ist. Auch hier wird nur überprüft, ob der Name des Problems leer ist oder ob es dieses Problem schon in der Datenbank gibt.

4.3 Backend und Datenbank

Die Kommunikation mit der Datenbank über das Backend, wurde in den vorherigen Unterkapiteln immer wieder angesprochen. Wie genau die Datenbank aufgebaut ist wurde allerdings noch nicht erläutert. Außerdem werden

auch die Schnittstellen aufgezählt, die das Backend zur Verfügung stellt, um auf die Datenbank zuzugreifen.

Zuerst wird die Datenbank behandelt. Die Datenbank wurde mit dem Datenbank Management System MongoDB [8] erstellt. Mit MongoDB können Daten als Sammlungen von Dokumenten, Collections genannt, abgespeichert werden. Die Dokumente sind JSON-ähnlich ([5], [9]), was heißt, dass mit Hilfe von MongoDB einfach Objekte eigener Klassen abgespeichert werden können. Welche Klasse die Objekte haben und wie die Klassen aufgebaut sind, die hier benutzt werden, ist gut am Klassendiagramm in Abbildung 66 zu erkennen. Im Klassendiagramm sind vier Klassen zu sehen, die so auch

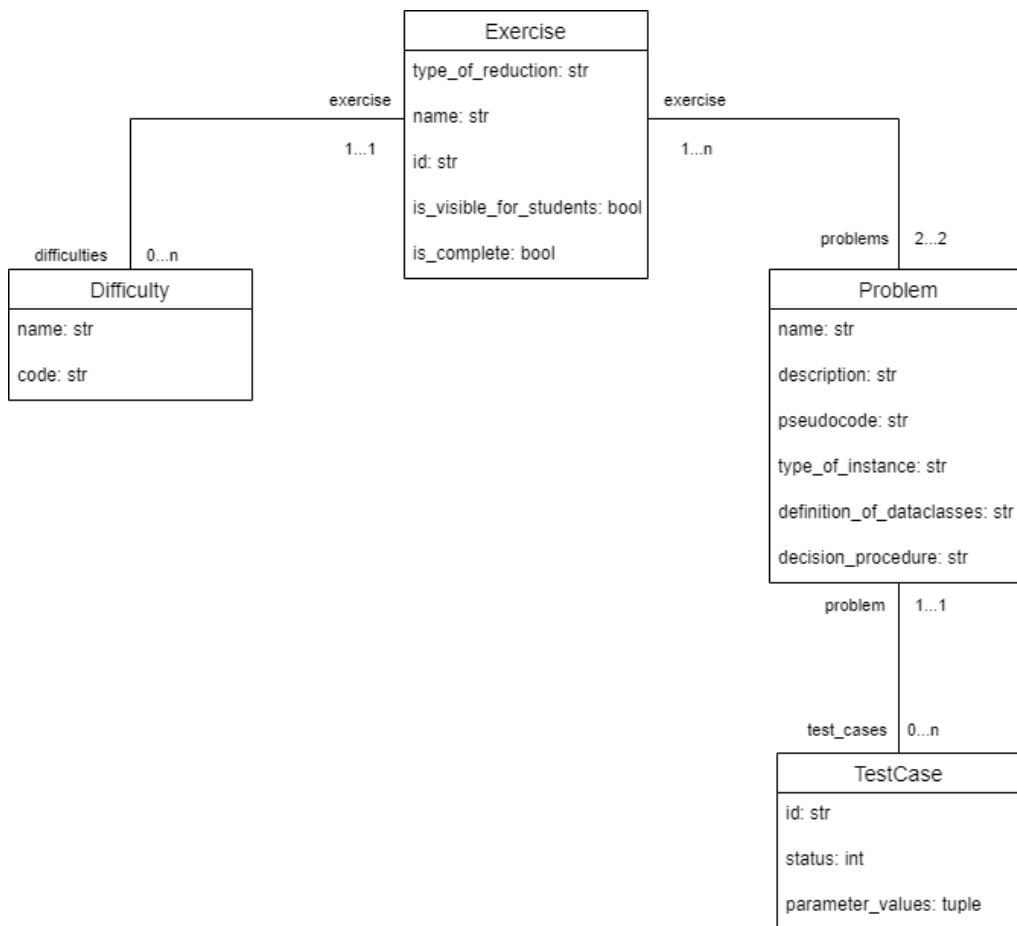


Abbildung 66: Klassendiagramm Reduktions-Trainer

im Backend angelegt wurden.

Attribute, welche nur indirekt bekannt sein dürften, werden kurz erläutert. `is_visible_for_students`, in der Klasse `Exercise`, bestimmt, ob die Übung für die Studierenden angezeigt wird. `is_complete`, welche auch in der Klasse `Exercise` ist, bestimmt, ob die Dozenten die Übung für die Studierenden sichtbar machen können. `status`, in der Klasse `TestCase`, bestimmt, zu was und ob ein Testfall ausgewertet wurde.

In der Datenbank wird für den Reduktions-Trainer eine Collection für Übungen, also Objekte der Klasse `Exercise` und eine Collection für Probleme, Objekte der Klasse `Problem` angelegt. Da Schwierigkeitsgrade, also Objekte der Klasse `Difficulty`, nur im Zusammenhang mit Übungen angelegt und ausgelesen werden braucht es dafür keine eigene Collection. Das Gleiche gilt für Testfälle, also Objekte der Klasse `TestCase`, mit Problemen.

Wird im Frontend ein neues Problem angelegt, muss dieses Problem nur in die Collection eingefügt werden. Ähnlich einfach funktioniert Überschreiben, Löschen und Laden, wobei dafür noch der Name des bereits in der Collection liegenden Problems mitgegeben werden muss. Wird im Frontend andererseits eine Übung angelegt, muss sowohl die Collection mit den Übungen als auch die mit den Problemen aktualisiert werden, da jede Übung zwei Probleme braucht.

Wie schon in Kapitel 3.4 beschrieben gibt es Routen über die mit dem Backend kommuniziert werden kann. Dabei werden Schnittstellen im Backend angesprochen, also einfach nur Methoden in Python Code. Es folgt eine Aufzählung aller Schnittstellen und wozu es sie gibt.

- **Alle Übungen aus der Datenbank laden.** Diese Schnittstelle wurde häufiger im Dozenten-Frontend angesprochen. Unter anderem wird sie dafür benutzt, um alle Übungen in der „Übungen verwalten“-Ansicht (Abb. 38) anzuzeigen.
- **Eine bestimmte Übung aus der Datenbank laden.** Dafür muss die Id der Übung mitgegeben werden, die geladen werden soll. Diese Schnittstelle wird benutzt, um die von den Studierenden ausgewählte Übung zu laden.
- **Eine Übung aus der Datenbank löschen.** Dafür muss die Id der zu löschenden Übung mitgegeben werden. Diese Schnittstelle wird von den Dozenten genutzt, wenn in der „Übungen verwalten“-Ansicht das Löschen einer Übung ausgewählt wird.

- **Eine Übung in der Datenbank anlegen.** Dafür muss eine Übung, die Information, ob Problem 1 aktualisiert werden soll und die Information, ob Problem 2 aktualisiert werden soll mitgegeben werden. Diese Schnittstelle wird in der „Übung erstellen“-Ansicht (Abb. 43) benutzt.
- **Eine Übung in der Datenbank überschreiben.** Dafür muss eine Übung (welche die Alte überschreiben soll), die Id der in der Datenbank gespeicherten Übung und ob für Problem 1 oder Problem 2 (oder beide) ein neues Problem erstellt werden soll mitgegeben werden. Diese Schnittstelle wird in der „Übung bearbeiten“-Ansicht (Abb. 61) benutzt.
- **Alle Probleme aus der Datenbank laden.** Diese Schnittstelle wurde häufiger im Dozenten-Frontend angesprochen. Unter anderem wird sie dafür benutzt, um alle Probleme in der „Entscheidungsprobleme verwalten“-Ansicht (Abb. 64) anzuzeigen.
- **Ein Problem aus der Datenbank löschen.** Dafür muss der Name des zu löschenden Problems mitgegeben werden. Diese Schnittstelle wird von den Dozenten genutzt, wenn in der „Entscheidungsprobleme verwalten“-Ansicht das Löschen eines Problems ausgewählt wird.
- **Ein Problem in der Datenbank anlegen.** Dafür muss ein Problem mitgegeben werden. Diese Schnittstelle wird in der „Problem erstellen“-Ansicht benutzt.
- **Ein Problem in der Datenbank überschreiben.** Dafür muss ein Problem (welches das Alte überschreiben soll) und der Name des in der Datenbank gespeicherten Problems mitgegeben werden. Diese Schnittstelle wird in der „Problem bearbeiten“-Ansicht benutzt.
- **Den Status eines Testfalls ändern.** Dafür muss der Name des Problems (zu welchem der Testfall gehört), die Id des Testfalls und der neue Status mitgegeben werden. Diese Schnittstelle wird ausschließlich vom Reduktionsverifizierer benutzt.
- **Sichtbarkeit einer Übung für Studierende ändern.** Dafür muss die Id der Übung und ob diese Übung für die Studenten sichtbar sein soll mitgegeben werden. Diese Schnittstelle wird in der „Übungen verwalten“-Ansicht benutzt, wenn die Checkbox zum Anzeigen einer Übung aktiviert oder deaktiviert wird.
- **Passwort überprüfen.** Dafür muss ein Passwort mitgegeben werden. Diese Schnittstelle wird bei der „Anmeldung“ benutzt (Abb. 35).

5 Fazit

Zusammengefasst ist der Reduktions-Trainer mit den Komponenten, die in dieser Arbeit hinzugefügt wurden, nun im Web benutzbar. Dozenten können Reduktionen und Entscheidungsprobleme in einer Datenbank abspeichern und diese für Studierende sichtbar machen. Studierende können Reduktions-Algorithmen selbst programmieren und werden vom Reduktionsverifizierer auf Fehler hingewiesen. Studierende können sogar einzelne Eingaben in den Reduktionscode testen und sehen das Ergebnis.

Der Reduktions-Trainer ist soweit schon auf einem guten Stand, dennoch gibt es Features und Funktionen, die ihn noch weiter verbessern würden. Die folgenden Features beziehen sich auf die Komponenten, die in dieser Ausarbeitung erläutert wurden, was den Reduktionsverifizierer ausschließt.

Wie in der „Trainer“-Ansicht zu sehen war, gibt der Reduktionsverifizierer englische Rückmeldungen (Abb. 23). Statt in Zukunft nur im Reduktionsverifizierer zu ändern, dass Rückmeldungen auch auf Deutsch sein können, könnte es im Frontend auch die Möglichkeit geben zwischen Deutsch und Englisch oder sogar noch anderen Sprachen zu wechseln.

Besonders praktisch wäre es auch, wenn die Beschreibung der Probleme von den Dozenten in LaTeX [6] angegeben werden könnte. Das würde die Beschreibung der Probleme für die Studierenden lesbarer machen.

In der Hoffnung, dass der Reduktions-Trainer in Zukunft begleitend zur Vorlesung eingesetzt wird und viele Studierende ihn benutzen, ist es ratsam die Sicherheit weiter zu erhöhen. Auf dem derzeitigen Stand wird das Frontend für die Dozenten nur mit einem Passwort geschützt, was im Backend ungeschützt liegt. Um die Sicherheit zu erhöhen, könnte implementiert werden, dass Dozenten sich erst registrieren müssen, um das Dozenten Frontend benutzen zu können. Das würde dafür sorgen, dass jeder/jede Dozent/Dozentin ein eigenes Passwort hat, welche alle hashed und somit unlesbar in der Datenbank liegen würden.

Das Frontend des Reduktions-Trainers hat also noch Potenzial für Verbesserungen, ist aber schon jetzt ein hilfreiches Tool, dass Studierende dabei unterstützen kann, das Anwenden von Reduktionen zu üben.

Literatur

- [1] *Angular*. <https://angular.io> (Jan. 2023).
- [2] Emma Enström. “On Difficult Topics in Theoretical Computer Science Education”. Diss. Stockholm, Schweden: KTH Royal Institute of Technology Computer Science und Communication, 2014.
- [3] Aris Filos-Ratsikas. “Why do Students Face Difficulties in Theoretical Computer Science? A Case Study”. In: (2021). University of Liverpool, United Kingdom.
- [4] *JavaScript*. <https://www.ecma-international.org> (Jan. 2023).
- [5] *JSON (JavaScript Object Notation)*. <https://www.json.org> (Jan. 2023).
- [6] *LaTeX – A document preparation system*. <https://www.latex-project.org> (Jan. 2023).
- [7] Kathrin Lehmann. “Toolgestütztes Lernen im Kontext der Berechenbarkeitstheorie: ein Backend-Prototyp zur automatisierten Überprüfung von Reduktionen”. In: (2021). Universität Kassel, FB 16, FG Theoretische Informatik / Formale Methoden.
- [8] *MongoDB*. <https://www.mongodb.com> (Jan. 2023).
- [9] *MongoDB JSON and BSON*. <https://www.mongodb.com/json-and-bson> (Jan. 2023).
- [10] *PrimeNG*. <https://www.primefaces.org/primeng> (Jan. 2023).
- [11] *Python*. <https://www.python.org> (Jan. 2023).
- [12] Boaz Barak Sanjeev Arora. *Computational Complexity: A Modern Approach*. Princeton University. Cambridge University Press, 2010.
- [13] Uwe Schöning. *Theoretische Informatik - kurz gefasst*. 5. Aufl. Heidelberg. Spektrum Akademischer Verlag, 2008.
- [14] *TypeScript*. <https://www.typescriptlang.org> (Jan. 2023).