UNIKASSEL
VERSITÄT

Fachbereich Elektrotechnik/Informatik
Fachgebiet Theoretische Informatik/Formale Methoden

Hypothesenüberprüfung in der Biologie: Erweiterung und Anbindung des Calculus of Influence an ein Lehr-/Lernsystem zur Überprüfung von Hypothesen

BACHELORARBEIT

zur Erlangung des Grades eines Bachelor of Science
im Fachbereich Elektrotechnik/Informatik
der Universität Kassel

Eingereicht von: Jan Heinemeyer
Matrikelnummer: [redacted]
E-Mail: [redacted]

Erstgutachter: Prof. Dr. Martin Lange
Zweitgutachterin: Prof. Dr. rer. nat. Claudia Fohry
Betreuer: Dr. Norbert Hundeshagen

Eingereicht am: 25. Juli 2023

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur mit den nach der Prüfungsordnung der Universität Kassel zulässigen Hilfsmitteln angefertigt habe. Die verwendete Literatur ist im Literaturverzeichnis angegeben. Wörtlich oder sinngemäß übernommene Inhalte habe ich als solche kenntlich gemacht. Die Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen. Die von mir eingereichte gedruckte Version meiner (schriftlichen) Arbeit stimmt mit der eingereichten digitalen Version überein.

Kassel, 25. Juli 2023

Jan Heinemeyer

Inhaltsverzeichnis

1 Einleitung

Im Rahmen der fortschreitenden Digitalisierung in der Lehre und des Bestrebens, die Qualität der Lehre zu verbessern, wurde das Projekt „Professionalisierung durch intelligente Lehr-/Lernsysteme“ (ProfiLL) [3] ins Leben gerufen. Es ist ein Teilprojekt des Forschungsprojekts „PRONET-D: Professionalisierung im Kasseler Digitalisierungsnetzwerk“[1], das am Zentrum für Lehrerbildung der Universität Kassel durchgeführt wird. Die vorliegende Arbeit ist Teil des ProfiLL-Projekts. Das übergeordnete Ziel von ProfiLL besteht darin, ein intelligentes Lehr-/Lernsystem (iLL) zu entwickeln, das individuelles und adaptives Lernen ermöglicht. Dabei wird untersucht, wie dieses System in eine hochschuldidaktische Lernumgebung integriert werden kann und inwiefern es die Ausbildung von Lehramtsstudierenden verbessern kann.

Zu diesem Zweck wurde ein iLL zur Überprüfung von Hypothesen von den Studierenden Georg Siebert, Stefan Kablowski und dem Autor dieser Arbeit im Rahmen ihrer HiWi-Tätigkeit und einer Projektarbeit am Fachgebiet Theoretische Informatik/Formale Methoden entwickelt. Das iLL ist speziell für den Einsatz in der Biologie-Lehre konzipiert und dient der Überprüfung von Hypothesen, die auf Grundlage eines beobachteten Phänomens und einer Forschungsfrage formuliert werden, um diese anschließend in einem Experiment zu verifizieren.
Die Überprüfung der Hypothese wird auf Basis einer formalen Grammatik durchgeführt, die beschreibt, wie eine Hypothese aufgebaut sein muss.

In dem Paper „Formal Reasoning about Influence in Natural Sciences Experiments“[1] und der Bachelorarbeit „An Efficient Algorithm for Proof Search in the Calculus of Influence“[5] wurde ein Kalkül entwickelt, das es ermöglicht, zu dem Datensatz eines Experiments und einem davon abgeleiteten *Influence Model* zu überprüfen, ob eine Hypothese korrekt ist. Ein Influence Model stellt eine Menge von Statements dar, wobei ein Statement eine abstrahierte Beschreibung eines Funktionsabschnitts darstellt. Die Überprüfung der Hypothese, die ebenfalls als Statement vorliegen muss, gelingt dann mit den Regeln des Kalküls.
Zusätzlich wurde im Rahmen der genannten Bachelorarbeit ein Python-Tool mit der Bezeichnung *Influence Solver* entwickelt, welches die Überprüfung einer Hypothese automatisiert durchführen kann.

In dieser Arbeit wird der Influence Solver in das iLL integriert, um die Überprüfung von Hypothesen nicht nur auf rein syntaktischer Ebene, sondern auch auf inhaltlicher bzw. semantischer Ebene zu ermöglichen. Dadurch ergeben sich zwei zentrale Forschungsfragen:

[1] https://www.uni-kassel.de/einrichtung/zlb/forschung-innovationsprojekte/pronet-d/startseite

F1) Wie lassen sich Hypothesen, die auf Basis der formalen Grammatik formuliert werden, in Statements überführen, hat dies Grenzen und ergeben sich aus der Arbeit mit dem Kalkül mögliche Erweiterungen der Hypothesen?

F2) Wie gestaltet sich das strukturierte Anlegen eines Experiments, um die für den Kalkül benötigten Daten zu erfassen?

Die Beantwortung dieser Fragen ist die Voraussetzung für eine erfolgreiche Integration des Influence Solvers in das iLL.

Im Folgenden wird ein Überblick über den Aufbau der Arbeit gegeben. Das Ziel des Kapitels 2 besteht darin, die notwendigen Grundlagen zu vermitteln. Es erklärt, wie sich die Durchführung eines Experiments in der Biologie gestaltet und gibt einen Überblick über das iLL. Des Weiteren werden in diesem Kapitel die formale Grammatik zur Überprüfung von Hypothesen sowie der Calculus of Influence vorgestellt.
Kapitel 3 stellt eine Funktion vor, mit der sich Hypothesen in Statements übersetzen lassen. Dabei wird auch auf die Grenzen der Übersetzung eingegangen. Das Kapitel schließt mit möglichen Erweiterungen, die die Anzahl der möglichen Arten von Hypothesen des iLL vergrößern könnten.
In Kapitel 4 wird die praktische Umsetzung der Integration beschrieben. Dabei wird zum einen die Umsetzung des Anlegens eines Experiments gezeigt, zum anderen wird ein Überblick über die praktische Umsetzung der Übersetzung gegeben die auf der Basis der Funktion aus Kapitel 3 umgesetzt worden ist. Auch hier werden Erweiterungsmöglichkeiten diskutiert.
In Kapitel 5 wird abschließend ein Fazit gezogen und ein Ausblick auf mögliche zukünftige Weiterentwicklungen des iLL gegeben.

2 Grundlagen

In diesem Kapitel wird zunächst das Konzept und die Durchführung eines Experiments in der Biologie-Lehre vorgestellt. Anschließend folgt eine Beschreibung einer Webanwendung, auf der diese Arbeit aufbaut. Zuletzt werden die theoretischen Grundlagen präsentiert, die für das Verständnis dieser Arbeit von Bedeutung sind.

2.1 Experimente in der Biologie

Das vorliegende Kapitel stellt eine Zusammenfassung der Projektarbeit „Categories of Biological Experiments“ von Lukas Mentel dar [4]. Es beschreibt den Ablauf der Durchführung eines Experiments in der Biologie. Diese Informationen sind wichtig für die vorliegende Arbeit, da das iLL in diesem Bereich eingesetzt wird.

In der Lehre der Biologie spielen Experimente und ihre Durchführung eine bedeutende Rolle. Durch sie sollen den Lernenden unter anderem Fähigkeiten vermittelt werden, wie sie zu einem beobachteten Phänomen eine fundierte Hypothese formulieren und diese dann kritisch überprüfen können, indem sie ein Experiment entwerfen und durchführen.

Die Durchführung eines Experiments im Rahmen einer Lehrveranstaltung folgt einer spezifischen Methodik, die aus mehreren Schritten oder Phasen besteht. Diese werden im Folgenden erklärt.

In der Biologie werden in der Regel Lebewesen und ihr Verhalten zu verschiedenen Einflüssen untersucht. Daher wird in einem ersten Schritt den Lernenden eine Spezies vorgestellt und dann lässt man sie ein bestimmtes Phänomen dieser Spezies beobachten. Dies dient dazu, dass die Lernenden ein intrinsisches Interesse an der Spezies und der Untersuchung des *Phänomens* erlangen.

Im nächsten Schritt wird eine *Forschungsfrage* formuliert. Sie dient zur Spezifizierung des zu untersuchenden Phänomens und zur Festlegung der erforderlichen Umweltfaktoren und Messparameter. Durch sie soll also ein Fokus für das weitere Vorgehen gesetzt werden und es werden die unabhängige und abhängige Variable definiert. Hierbei stellen die Umweltfaktoren und Messparameter die *unabhängige Variable* und die Eigenschaft oder das Merkmal, welche bei der Spezies beobachtet wurden, die *abhängige Variable* dar. Sie stellt also die Frage nach dem Zusammenhang zwischen den Umweltfaktoren und dem Verhalten der Spezies bzw. zwischen den beiden Variablen. Eine Forschungsfrage könnte zum Beispiel lauten: „Wie hängt die Aktivität der Hefe mit der Temperatur zusammen?“.

Als Nächstes werden Annahmen über die Verhaltensänderungen der Spezies in Abhängigkeit zu den sich veränderten Umweltfaktoren gebildet. Diese Annahmen werden auch

Hypothesen genannt. Eine Hypothese beschreibt hierbei eine Veränderung der Umweltfaktoren, also der unabhängigen Variable, und die erwartete Veränderung der Eigenschaft bzw. des Verhaltens der Spezies, also die Veränderung der abhängigen Variable, aufgrund dessen. Beispiel einer Hypothese: “Eine Erhöhung der Temperatur steigert das Hefewachstum”.

Wurde eine Hypothese gebildet, geht es mit der Konzeptphase weiter. In ihr wird in Abhängigkeit einer Hypothese und deren Variablen eine Schritt-für-Schritt-Anleitung erstellt, wie das Experiment durchgeführt wird und welche Materialien dafür benötigt werden.

Im Anschluss daran wird mit der Durchführung des Experiments begonnen. Ziel ist hierbei die Überprüfung der zuvor aufgestellten Hypothesen. Hierfür sind mehrere Durchführungen notwendig, wobei die Beobachtungen und Messungen eines jeden Durchgangs festgehalten werden.

Anschließend werden die Beobachtungen und Messungen ausgewertet und überprüft, ob sie die Hypothese stützen. Sollten die Ergebnisse der Experimente die Hypothese nicht stützen, gilt es eine genaue Fehleranalyse durchzuführen, um herauszufinden, ob die Hypothese falsch ist oder ob es am Experimentaufbau liegt.

Ist das Ergebnis der Fehleranalyse, dass die Hypothese falsch ist, wird eine neue formuliert und der Prozess erneut durchgeführt.

2.2 Bisherige Automatisierung der Hypothesenüberprüfung

Wie bereits erwähnt, wurde im Rahmen des ProfiLL Projekts [3] ein iLL[1] entwickelt, welches Lernenden vermittelt, wie Hypothesen im Kontext von biologischen Experimenten formuliert werden.

Das Lehr-Lernsystem zur Überprüfung von Hypothesen ist eine Webanwendung, die zwei Oberflächen umfasst. Eine Oberfläche für Dozierende, deren Hauptfeature das Anlegen von Experimenten ist. Die zweite Oberfläche ist für die Lernenden. In ihr kann zu einem vorhandenen Experiment Hypothesen formuliert und überprüft werden.

Die Oberfläche zum Erstellen von Experimenten besteht aus einem Stepper[2] mit vier Schritten. Im ersten Schritt werden die allgemeinen Informationen angelegt. Diese sind ein Titel und eine Beschreibung für das Experiment, die in sämtlichen Übersichten zur Identifikation eines Experiments dienen. Im zweiten Schritt wird die Forschungsfrage in einer Freitexteingabe erstellt. Als Drittes können dann Wörter erstellt werden. Hierfür wählt man einen der Worttypen aus, gibt den Bezeichner und gegebenenfalls Zusatzinformationen an und fügt das Wort dann der Liste von Wörtern hinzu. Die erstellten Wörter werden später in der Lernenden-Oberfläche als Wortbausteine dargestellt, mit denen die Lernenden Hypothesen erstellen können. Abbildung 2.1 zeigt diesen Schritt. Die

[1] https://syre.fm.cs.uni-kassel.de/Georg/bio-tool/-/tree/maste
[2] https://material.angular.io/components/stepper/overview

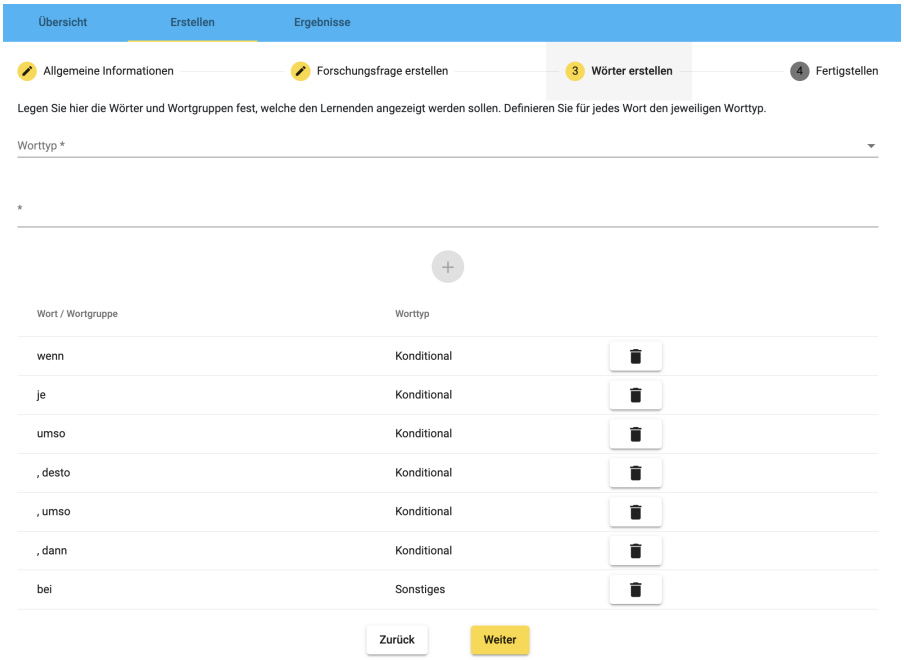


Abbildung 2.1: Schritt 3: Erstellen von Wörtern - Ursprüngliche Benutzeroberfläche zum Anlegen eines Experimentes

Auswahl eines Typs bei der Erstellung eines Wortes ist notwendig für die interne Logik zur Überprüfung einer Hypothese, auf die im nächsten Kapitel im Detail eingegangen wird. Der letzte Schritt stellt eine Übersicht dar, in der alle zuvor eingegebenen Daten noch mal überprüft werden können, bevor das Experiment erstellt wird. Die Oberfläche bietet ebenfalls eine Übersicht über alle bereits erstellten Experimente und die Möglichkeit, diese zu bearbeiten, sowie eine Download-Funktion, mit der die erstellten Hypothesen der Lernenden abgerufen werden können.

Die zweite Oberfläche dient dem Erstellen und Überprüfen von Hypothesen. Sie bietet eine Übersicht über alle erstellten Experimente, von denen eins gestartet werden kann. Als Erstes muss eine ID eingegeben werden. Die ID dient dazu, dass beim Download in der ersten Oberfläche Daten gebündelt zu einer ID ausgegeben werden. In der nächsten Ansicht wird die Forschungsfrage zu dem ausgewählten Experiment angezeigt und die Aufforderung, mindestens zwei Hypothesen zu dieser zu formulieren. Hierfür wird ein Freitextfeld zur Verfügung gestellt, in dem eine Hypothese eingegeben werden kann und mit einem Button zu einer Liste hinzugefügt werden kann. Es bietet so die Möglichkeit, eine unbegrenzte Anzahl an Hypothesen zu formulieren. Über einen „Weiter“-Button wird zur nächsten Ansicht navigiert. In dieser Ansicht wird wieder die Forschungsfrage angezeigt und erneut die Aufforderung, mindestens zwei Hypothesen zu erstellen. In dieser Ansicht gibt es jedoch kein Freitextfeld, sondern einen Kasten, in dem die für dieses Experiment erstellten Wörter in Form von Wortbausteinen dargestellt werden. Um eine Hypothese zu erstellen, bietet das Tool einen zweiten Kasten, in den die Wortbausteine per Klick oder Drag-and-drop bewegt werden können, ebenso wie einen Button zum Überprüfen und einen zum Löschen der Hypothese. Abbildung 2.2 zeigt diese Oberfläche. Wurde eine Hypothese erstellt und auf Überprüfen geklickt, öffnet sich ein Dialog. Es gibt drei verschiedene Dialoge. „Die Hypothese passt nicht zur Forschungsfrage“ wird

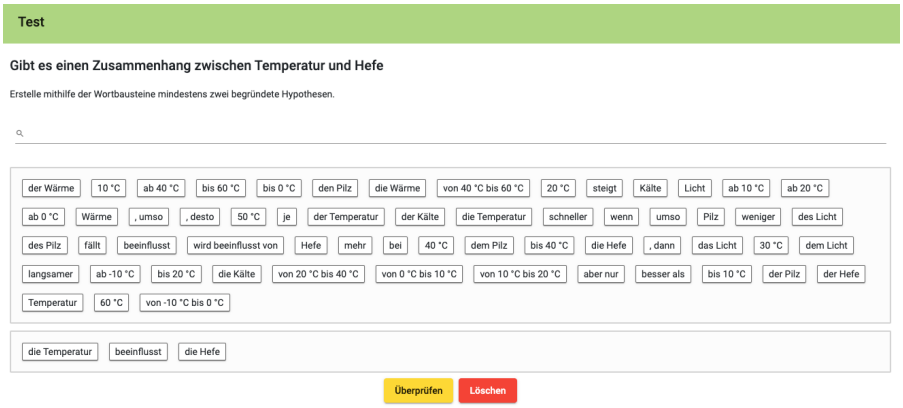


Abbildung 2.2: Benutzeroberfläche zur Formulierung und Überprüfung von Hypothesen.

angezeigt, wenn eine Variable verwendet wurde, die nicht Teil der Forschungsfrage ist. Ist die Hypothese fehlerhaft, erscheint der Dialog „Die Hypothese ist nicht überprüfbar“. Wurde eine syntaktisch korrekte Hypothese erstellt, erscheint der Dialog „Die Hypothese ist überprüfbar“ mit der zusätzlichen Aufforderung, eine Begründung in ein Freitextfeld einzugeben. In jedem Fall lässt sich der Dialog mit dem OK-Button schließen. Im letzten Fall wird die Hypothese einer Liste hinzugefügt. Über der Liste befindet sich die Aufforderung, eine der Hypothesen aus der Liste auszuwählen, welche in einem Experiment überprüft werden soll. Wurde eine Hypothese ausgewählt und auf den Weiter-Button geklickt, fordert die nächste Ansicht auf, mit der Planung für das Experiment zu beginnen, welches ausgewählt wurde.

2.3 Theoretische Grundlagen

Die theoretische Basis dieser Arbeit bilden eine Grammatik, die den Aufbau von Hypothesen beschreibt, und der *Calculus of Influence*.[1][5] Die Grammatik wird in dem iLL genutzt, um die Hypothesen syntaktisch zu überprüfen und der Kalkül kann Hypothesen semantisch überprüfen. Beides wird in diesem Kapitel vorgestellt.

Formale Grammatik für die Hypothesenbildung

Bevor wir die im iLL verwendete Grammatik vorstellen, wollen wir zunächst definieren, was eine Grammatik ist. Die folgenden Definitionen zu den Grundlagen einer Grammatik beruhen auf dem Buch „Theoretische Informatik-kurz gefasst“ entnommen.[7]

Definition 2.3.1 (Alphabet, Wort) *Ein Alphabet Σ ist eine nicht leere, endliche Menge von Symbolen. Ein Wort w über Σ ist eine endliche Folge von Symbolen $a_0a_1...a_n$ wobei gilt $a_i \in \Sigma$ für $i \in \{0, 1, ..., n\}$. Σ^* ist dann die Menge aller Wörter über Σ.*

Ein Alphabet ist eine Menge von Symbolen, aus denen durch Aneinanderreihung Wörter gebildet werden können. Wenn zum Beispiel $\Sigma = \{a, b\}$ ist, dann ist das Wort $w = abba$ ein mögliches Wort über dem Alphabet Σ.

Definition 2.3.2 (Eine Grammatik) *Eine Grammatik ist ein 4er-Tupel $G = (V, \Sigma, P, S)$ und es gelten die folgenden Bedingungen. V ist die endliche Menge der Nichtterminale. Σ ist die endliche Menge der Terminale. Der Schnitt der Nichtterminale und Terminale muss leer sein. P ist die endliche Menge der Produktionsregeln. Sie ist eine endliche Teilmenge von $(V \cup \Sigma)^+ \times (V \cup \Sigma)^*$. $S \in V$ ist das Startsymbol.*

Eine Grammatik stellt eine Menge von Regeln dar, die beschreiben, wie Wörter über einem Alphabet durch die Verwendung von Produktionsregeln ausgehend von einem Startsymbol erzeugt werden können. Ein Beispiel für eine Grammatik ist $G = (\{S\}, \{a, b\}, \{(S, SS), (S, a), (S, b)\}, S)$. In dieser Grammatik gibt es ein Nichtterminalsymbol S, das gleichzeitig das Startsymbol ist. Das Terminalalphabet besteht aus den Symbolen a und b, und die Produktionsregeln besagen, dass ein S durch SS, a oder b ersetzt werden kann.

Definition 2.3.3 (Ableitungsrelation) *Seien $u, v \in (V \cup \Sigma)^*$ und G eine Grammatik. Dann ist die Relation $u \Rightarrow_G v$ wie folgt definiert: Hat u die Form xyz, dann hat v die Form $xy'z$ falls $x, z \in (V \cup \Sigma)^*$ und $(y, y') \in P$. Des Weiteren sei $u \Rightarrow_G^* v$ die reflexive und transitive Hülle von $u \Rightarrow_G v$.*

Eine Ableitungsrelation beschreibt, wie ein Wort unter Verwendung der Regeln einer Grammatik in ein anderes Wort abgeleitet werden kann. Zum Beispiel kann die Ableitung $aSSb \Rightarrow_B aaSb$ gemäß der vorherigen Grammatik B durchgeführt werden. Die Definition $u \Rightarrow_G^* v$ ist nützlich, um lange Ableitungsketten zu verkürzen. Dadurch können wir beispielsweise schreiben, dass $aSSb \Rightarrow_B^* aabb$ gilt, ohne alle Zwischenschritte detailliert auflisten zu müssen.

Definition 2.3.4 (Sprache, Sprache einer Grammatik) *Die Menge L heißt Sprache, wenn $L \subseteq \Sigma^*$. Die Sprache einer Grammatik G mit Terminalalphabet Σ und Startsymbol S, wird definiert als $L(G) = \{w \in \Sigma \mid S \Rightarrow_G^* w\}$.*

Eine Sprache ist also eine Menge von Wörtern über einem Alphabet. Die Sprache einer Grammatik G sind alle Wörter, die sich unter Berücksichtigung der Regeln von G ableiten lassen und nur noch Terminalsymbole enthalten. Für die Grammatik B sind das alle Wörter, die ausschließlich aus a's und b's bestehen ($L(B) = \{a, b\}^+$).

Im Rahmen des iLL zur Überprüfung von Hypothesen wurde eine Grammatik entworfen, die festlegt, wie eine Hypothese strukturiert sein soll. Mithilfe dieser Grammatik wird die Überprüfung durchgeführt. Im Folgenden wird die Grammatik anhand ihrer Produktionsregeln vorgestellt. Dabei sind alle Symbole, die mit Kleinbuchstaben beginnen, *Nichtterminale*, während jene, die mit Großbuchstaben starten, Terminalsymbole darstellen. Anstelle von Tupeln verwenden wir eine Pfeilschreibweise. Zum Beispiel wird (A, B), (A, C) als $A \to B \mid C$ dargestellt. Die Kurzschreibweise $A \to BC?$ steht für $A \to B \mid BC$. Die vollständige Grammatik ist im Anhang (A.1) zu finden.

Bevor die Grammatik vorgestellt wird, ist es wichtig, eine Besonderheit hervorzuheben: Die Terminalsymbole der Grammatik fungieren als sogenannte Tokens. Ähnlich wie

bei einem Lexer bestehen diese Tokens aus einem Typ und einem Wert. Die in der Grammatik verwendeten Terminalsymbole entsprechen den Worttypen, während der Wert Wörter oder Wortbausteine aus der deutschen Sprache sein können, je nachdem, wie das zugrunde liegende Experiment definiert wurde. Ein Beispiel für einen Worttypen ist UnspezifischeVariable und ein Wortbaustein kann ein einzelnes Wort oder eine Gruppe von Wörtern sein wie zum Beispiel „wird beeinflusst von“.
Die hier verwendeten Tokens besitzen noch drei weitere Werte, die noch zusätzliche Informationen enthalten. In der nachfolgenden Definition werden die möglichen Werte, die sie annehmen können, aufgezeigt, jedoch nicht ihre Bedeutung. Diese folgt in Kapitel 3 mit ausführlichen Erklärungen. Im Folgenden spielen sie nur eine untergeordnete Rolle.

Definition 2.3.5 (Token) *Ein Token ist ein 5-Tupel der Form (t, s, d, b, i), wobei t (wie type) den Worttypen, s (wie string) das Wort oder den Wortbaustein, $d \in \{\triangleleft, \triangleright\}$ (wie direction) eine Richtungsangabe, $b \in \{\rightarrow, \nearrow, \searrow\!\!\rightsquigarrow\}$ (wie behavior) ein Verhalten und i ein Intervall angibt. Mit Ausnahme von t, s können alle Attribute auch den Wert _ annehmen.*

Der Token (UnspezifischeVariable, *die Temperatur*, _, _, _) ist ein Beispiel für eine unspezifische Variable mit dem Wortbaustein „die Temperatur“. Da eine Variable keine Richtung und kein Monotonieverhalten hat, wird für diese beiden Attribute der Wert _ genutzt, mit der Bedeutung, dass diese Werte für dieses Token nicht gesetzt sind oder ignoriert werden können. Ein weiteres Beispiel ist (UnspezifischeAbhängigkeit, beeinflusst, $\triangleright$, $\rightsquigarrow$, _). Nachfolgend werden in der Grammatik aus Lesbarkeitsgründen die Tokens auf ihre Worttypen reduziert.

Hypothesen werden in einfache Zusammenhänge und komplexe Zusammenhänge unterteilt. Diese Klassifizierung wurde für ein Bewertungsschema entwickelt, das jedoch noch nicht in das iLL implementiert wurde. Die Unterteilung ist in der Grammatik durch die folgende Startregel dargestellt:

$$\textit{hypothese} \rightarrow \textit{einfacherZusammenhang} \mid \textit{komplexerZusammenhang}$$

Bei der Beschreibung eines einfachen Zusammenhangs werden zusätzlich drei Satzformen unterschieden, die sich in ihrem Aufbau stark voneinander unterscheiden. Aus diesem Grund wurden sie durch zusätzliche Produktionsregeln klassifiziert:

$$\textit{einfacherZusammenhang} \rightarrow \textit{unspezifischerSatz} \mid \textit{spezifischerSatz} \mid \textit{konditionalSatz}$$

Ein unspezifischer Satz beschreibt einen allgemeinen Zusammenhang zwischen zwei unspezifischen Variablen. Ein Beispiel für eine solche Hypothese ist „Die Temperatur beeinflusst die Hefeaktivität“. In diesem Beispiel sind „Hefeaktivität“ und „Temperatur“ die unspezifischen Variablen und „beeinflusst“ ist die unspezifische Abhängigkeit.

unspezifischerSatz →
UnspezifischeVariable UnspezifischeAbhängigkeit UnspezfischeVariable

Im spezifischen Satz werden spezifische Variablen, spezifische Abhängigkeiten, temporale und quantitative Angaben verwendet, um die Zusammenhänge der Variablen genauer zu beschreiben. Der Unterschied zwischen unspezifischen und spezifischen Variablen besteht darin, dass eine spezifische Variable einen konkreten Wert beschreibt, während eine unspezifische Variable lediglich einen allgemeinen Namen für eine Variable darstellt.
Ein Beispiel für einen spezifischen Satz lautet „Der Pizzateig steigt bei 20 Grad“. In diesem Beispiel beschreibt die spezifische Abhängigkeit „steigt“ den genaueren Zusammenhang zwischen den Variablen im Vergleich zur unspezifischen Abhängigkeit „beeinflusst“. „20 Grad“ ist hier die spezifische Variable, während „Temperatur“ eine dazu passende unspezifische Variable sein kann. Durch den Einsatz von temporalen und quantitativen Angaben kann die Beeinflussung der unabhängigen Variable auf die abhängige Variable genauer beschrieben werden, indem Wörter wie „schneller“ (Temporal) oder „mehr“ (Quantität) verwendet werden.

Die Unterteilung eines spezifischen Satzes in „abhängigeVariableSpezifischeAbhängigkeit“ und „spezifischeAbhängigkeitAbhängigeVariable“ dient dazu, in den folgenden Ableitungsregeln mit zwei unspezifischen Variablen festzulegen, an welcher Stelle die abhängige Variable platziert sein sollte. In einem Fall steht die spezifische Abhängigkeit direkt nach der abhängigen Variable, im anderen Fall davor. Dies ist sowohl für das Fehlerfeedback als auch für die Übersetzung in Statements, die in einem späteren Kapitel erklärt werden, von Bedeutung.

spezifischerSatz →
abhängigeVariableSpezifischeAbhängigkeit | *spezifischeAbhängigkeitAbhängigeVariable*
abhängigeVariableSpezifischeAbhängigkeit →
UnspezifischeVariable SpezifischeAbhängigkeit Temporal? Bei SpezifischeVariable |
UnspezifischeVariable SpezifischeAbhängigkeit Temporal? Bei Quantität?
UnspezifischeVariable Temporal?
spezifischeAbhängigkeitAbhängigeVariable →
Bei SpezifischeVariable SpezifischeAbhängigkeit UnspezifischeVariable Temporal? |
Bei Quantität? UnspezifischeVariable SpezifischeAbhängigkeit UnspezifischeVariable

Mit einem Konditionalsatz können Sätze gebildet werden, um Bedingungen und ihre Konsequenzen auszudrücken oder eine proportionale Beziehung zwischen den Variablen zu beschreiben. Dabei gibt es zwei Formen.
Die erste Form folgt dem Muster „Wenn ..., dann ...“. Sie beschreibt den Zusammenhang zwischen den Variablen anhand einer Veränderung der unabhängigen Variable und ihrer Auswirkung auf die abhängige Variable. Ein Beispiel für einen solchen Satz ist „Wenn die Temperatur steigt, dann steigt die Hefeaktivität“.
Die andere Form verwendet die Formulierung „Je ..., desto ...“. Sie beschreibt eine proportionale Beziehung zwischen den Variablen. Ein Beispiel dafür ist „Je mehr die Temperatur steigt, desto mehr steigt der Pizzateig“.
Optional können beide Sätze durch temporale oder quantitative Angaben weiter konkretisiert werden.

konditionalSatz →
Konditional UnspezifischeVariable (Quantität | Temporal)? SpezifischeAbhängigkeit
Konditional SpezifischeAbhängigkeit UnspezifischeVariable (Quantität|Temporal)? |
Konditional (Quantität|Temporal)? UnspezifischeVariable SpezifischeAbhängigkeit
Konditional (Quantität|Temporal)? SpezifischeAbhängigkeit UnspezifischeVariable

Bei der Beschreibung eines komplexen Zusammenhangs werden zwei Satzformen unterschieden. Zum einen gibt es den Vergleichssatz und zum anderen eine Satzform, die aus dem einfachen Zusammenhang in Kombination mit einer Einschränkung besteht.

komplexerZusammenhang →
vergleichSatz | *einfacherZusammenhang einschränkungsSatz*

Eine Hypothese kann auch in Form eines Vergleichs formuliert werden. Dafür stehen zwei Formen zur Verfügung.
Die erste Form ermöglicht allgemeinere Formulierungen wie zum Beispiel den Satz „Mehr Temperatur beeinflusst die Hefeaktivität besser als weniger Temperatur“. In diesem Fall dient die Wortgruppe „besser als“ als Vergleichsoperator. Mit dieser Form kann beschrieben werden, wie sich die Qualität (VergleichsOperator) des einfachen Zusammenhangs durch Veränderungen der Quantität der unabhängigen Variable verändert.
Die zweite Form ermöglicht spezifischere Vergleiche. Ein Beispiel für einen solchen Satz lautet „Die Hefeaktivität steigt bei 20 Grad besser als bei 15 Grad“. Hier wird eine spezifische Abhängigkeit und eine spezifische Variable für die unabhängige Variable verwendet, um einen präzisen Vergleich zu formulieren.

vergleichSatz →
Quantität UnspezifischeVariable UnspezifischeAbhängigkeit UnspezifischeVariable
VergleichsOperator Quantität UnspezifischeVariable |
UnspezifischeVariable SpezifischeAbhängigkeit Bei SpezifischeVariable
VergleichsOperator Bei SpezifischeVariable

Eine weitere Möglichkeit, einen komplexen Zusammenhang zu formulieren, besteht darin, eine Satzform des einfachen Zusammenhangs mit einer Einschränkung zu kombinieren. Hierbei wird der unspezifische Satz des einfachen Zusammenhangs mit der Einschränkung verbunden, um den komplexen Zusammenhang zu formulieren. Ein Beispiel hierfür wäre „Die Temperatur beeinflusst die Hefe, aber nur zwischen 20°C und 50°C“.
Die Einschränkung besteht aus einem Konnektor, der den einfachen Zusammenhang mit der Einschränkung verbindet, zum Beispiel „aber nur“ sowie einem Wertebereich, der die Einschränkung präzise beschreibt, wie in diesem Fall „zwischen 20°C und 50°C“. Weitere

Beispiele für solche Wertebereiche sind „ab 20°C“ oder „bis 50°C“.
Die folgende Produktionsregel beschreibt eine Einschränkung.

$$\textit{einschränkung} \rightarrow \text{Konnektor Wertebereich}$$

Der Calculus of Influence

Der *Calculus of Influnece* wurde in dem Paper „Formal Reasoning about Influence in Natural Sciences Experiments“ [1] und der Bachelorarbeit „An Efficient Algorithm for Proof Search in the Calculus of Influence“ [5] erarbeitet und wird auf Basis dieser beiden Arbeiten hier vorgestellt.

Der *Calculus of Influence* wurde, wie bereits erwähnt, auch im Rahmen des Projekts ProfiLL entwickelt. Die Ausarbeitung des Kalküls dient als theoretische Grundlage für einen Algorithmus, der Hypothesen aus semantischer Sicht überprüfen kann. Im Rahmen der Bachelorarbeit [5] wurde auch eine Implementation des Algorithmus entwickelt, die im weiteren Verlauf dieser Arbeit als *Influence Solver* bezeichnet wird. Im Folgenden wird erst eine mathematische Abstraktion für Experimente vorgestellt und anschließend der Kalkül.

Im Folgenden sei $\mathbb{R}_\infty$ definiert als $\mathbb{R} \cup \{-\infty, \infty\}$ und $\leq_{\mathbb{R}_\infty}$ als die Erweiterung von $\leq_{\mathbb{R}}$ um die Relationen $\{(x, \infty) | x \in \mathbb{R}\}$ und $\{(-\infty, x) | x \in \mathbb{R}\}$ auf $\mathbb{R}_\infty$. Diese Definition entspricht der ursprünglichen Kleiner-gleich-Ordnung auf den reellen Zahlen jedoch mit der Hinzunahme eines Minimums und Maximums. Des Weiteren sein $a, b \in \mathbb{R}$ und die folgenden Intervalle als Teilmengen von $\mathbb{R}$ definiert:

$$\begin{aligned}
[a, b] &= \{x \mid a \leq x \leq b\} && \text{(geschlossenes Intervall)} \\
[-\infty, b] &= \{x \mid -\infty < x \leq b\} && \text{(links-offenes Intervall)} \\
[a, \infty] &= \{x \mid -\infty < x \leq b\} && \text{(rechts-offenes Intervall)} \\
[-\infty, \infty] &= \{x \mid -\infty < x \leq b\} && \text{(offenes Intervall)}
\end{aligned}$$

Sei im Folgenden $\mathcal{V} = \{a, b, ...\}$ eine endliche Menge von Variablen, die teilweise durch $\leq$ geordnet sind.

Definition 2.3.6 ($\mathcal{V}$-Statement) *Ein $\mathcal{V}$-Statement ist ein 5-Tupel der Form (a, I_1, q, I_2, b), wobei $a, b \in \mathcal{V}$ sind, I_1, I_2 Intervalle über $\mathbb{R}$ darstellen und q das Verhalten des Statements angibt. Das Verhalten q kann einen der folgenden Werte annehmen: monoton steigend ($\nearrow$), monoton fallend ($\searrow$), konstant ($\rightarrow$) oder willkürlich ($\rightsquigarrow$). Wenn der Kontext eindeutig ist, kann auch einfach von einem Statement gesprochen werden.*

Ein solches Statement beschreibt, dass eine Variable a im Intervall I_1 mit der Qualität q die Variable b so beeinflusst, dass b Werte aus dem Intervall I_2 annimmt. Ein Statement beschreibt also einen Ausschnitt einer Funktionsschar. So ist I_1 als ein Ausschnitt des Definitionsbereichs zu betrachten und I_2 als ein Ausschnitt des Wertebereichs. Das Verhalten q legt fest welches Monotonieverhalten in dem Ausschnitt vorliegen muss. Die vier möglichen Verhaltensformen eines Statements unterliegen der folgenden Ordnung: $\rightarrow \preceq q \preceq \rightsquigarrow$ mit $q \in \{\nearrow, \searrow\}$.

Definition 2.3.7 (Influence Model) *Ein* $\mathcal{V}$-Influence Model $\mathcal{M}$ *ist eine endliche Menge von* $\mathcal{V}$*-Statements. Dabei kann, wenn der Kontext eindeutig ist, auf die explizite Erwähnung von* $\mathcal{V}$ *verzichtet werden.*

Mit der Definition des *Influence Models* liegt nun eine mathematische Abstraktion eines Experiments vor. Ein *Influence Model*, das ausschließlich a, b-Statements enthält und sich auf zwei Variablen beschränkt, kann in einem zweidimensionalen Koordinatensystem dargestellt werden. Hierbei werden die Statements als Rechtecke visualisiert, wobei die Intervalle zur Bestimmung von Breite und Höhe verwendet werden. Für ein Statement (a, I_0, q, I_1, b) entspricht I_0 der Breite und I_1 der Höhe des Rechtecks. In Abbildung 2.3 ist eine grafische Darstellung eines *Influence Model*s zu sehen.

Definition 2.3.8 (Influence Exerperiment) *Die Funktion* $\mathcal{F} : \mathcal{V}^2 \to (\mathbb{R} \to \mathbb{R})$ *heißt Influence Experiment.*

Eine Funktion $\mathcal{F}(a, b)$ mit $a, b \in \mathcal{V}$ beschreibt den Einfluss der Variable a auf die Variable b, wobei $a \leq b$ gelten muss. Einflüsse verhalten sich transitiv und $\mathcal{F}(a, a)$ verhält sich wie die Identitätsfunktion.

Für ein Influence Experiment $\mathcal{F}(a, b)$ und das Statement $S = (a, [x, y], q, [x', y'], b)$ wird q aus einem der folgenden Fälle gebildet:

- wenn f.a. $z, z' \in [x, y]$ mit $z \leq z'$ und $\mathcal{F}(a, b)(z) \leq \mathcal{F}(a, b)(z')$ gilt, dann ist $q = \nearrow$
- wenn f.a. $z, z' \in [x, y]$ mit $z \leq z'$ und $\mathcal{F}(a, b)(z') \leq \mathcal{F}(a, b)(z)$ gilt, dann ist $q = \searrow$
- wenn f.a. $z, z' \in [x, y]$ mit $\mathcal{F}(a, b)(z') \leq \mathcal{F}(a, b)(z')$ gilt, dann ist $q = \to$
- falls keiner der zuvor genannten Fälle zutrifft, ist $q = \rightsquigarrow$

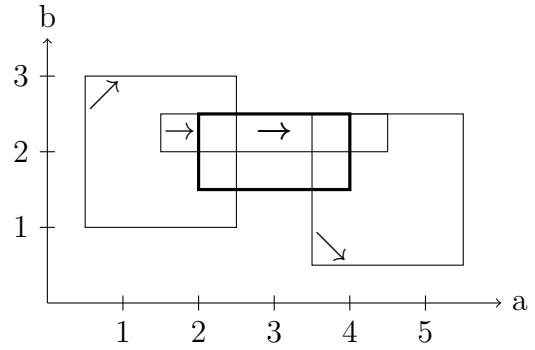
Alls letztes gilt für alle $z \in [x, y]$, dass $\mathcal{F}(a, b)(z) \in [x', y']$.

Der Einfluss $\mathcal{F}(a, b)$ erfüllt das Statement S genau dann, wenn für alle $z \in [x, y]$ gilt, dass $\mathcal{F}(a, b)(z) \in [x', y']$ und wenn das Monotonieverhalten von $\mathcal{F}(a, b)$ im Intervall $[x, y]$ monoton steigend ist und $q = \nearrow$ oder monoton fallend und $q = \searrow$ oder konstant und $q = \to$ oder $q = \rightsquigarrow$.

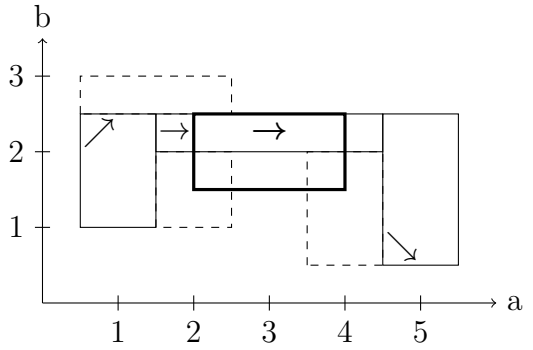
Ein Influence Experiment F erfüllt nun wiederum ein $\mathcal{V}$-Statement $S = (a, [x, y], q, [x', y'], b)$, wenn $F(a, b)$ S erfüllt und F erfüllt ein $\mathcal{V}$-Influence Model M, wenn es alle $S \in M$ erfüllt.

Definition 2.3.9 (Stärkeres Statement) *Sein* $S_1 = (a, I_1, q, I_2, b)$ *und* $S_2 = (a, I_1', q', I_2', b)$ *zwei Statements.* S_1 *ist stärker als* S_2*, wenn gilt:* $I_1' \subseteq I_1$, $I_2 \subseteq I_2'$ *und* $q \preceq q'$.

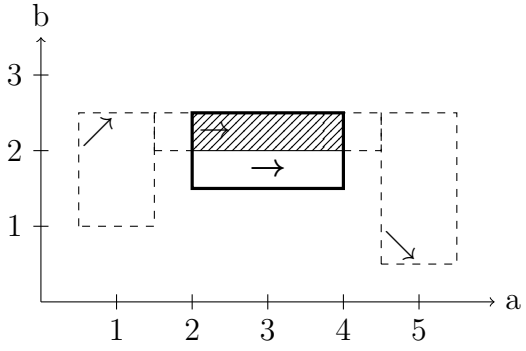
Mit dem *Calculus of Influence* kann nun überprüft werden, ob das Verhalten, das durch ein $\mathcal{V}$-Statement H beschrieben wird, dem eines zugrunde liegenden $\mathcal{V}$*-Influence Models* $\mathcal{M}$ entspricht. Hierfür bietet der Kalkül eine Reihe von Regeln (vgl. [5], S. 6), mit denen geprüft werden kann, ob sich das Statement aus dem Modell ableiten lässt. Ableiten bedeutet in diesem Kontext, dass versucht wird, durch Anwendung der Regeln auf die Statements von $\mathcal{M}$ ein stärkeres Statement zu erzeugen als H.



(a) Die drei Statements aus $\mathcal{M}$ (dünn) und H (dick).



(b) H und $\mathcal{M}$ nach Normalisierung mit entfernten Bereichen (gestrichelt).



(c) H (dick) entfernte Bereiche (gestrichelt) Statement, das stärker als H ist (diagonale Linien).

Abbildung 2.3: In drei Schritten zu einem stärkeren Statement als die Hypothese.

Abbildung 2.3 zeigt grafisch in drei Schritten, wie für das Statement $H = (a, [2, 4], \rightarrow, [1.5, 2.5], b)$ ein stärkeres Statement in $\mathcal{M} = \{A = (a, [0.5, 2.5], \nearrow, [1, 3], b), B = (a, [1.5, 4.5], \rightarrow, [2, 2.5], b), C = (a, [3.5, 5.5], \searrow, [0.5, 2.5], b)\}$ gefunden wurde.
Die Ausgangssituation ist in Abbildung 2.3a zu sehen. Die drei Statements aus $\mathcal{M}$ sind mit dünnen Linien dargestellt, während Statement H mit dicken Linien gezeichnet wurde. Die Pfeile in der grafischen Darstellung der Statements repräsentieren das Monotonieverhalten. Abbildung 2.3b zeigt $\mathcal{M}$, nachdem es normalisiert wurde. Ein **normalisiertes** *Influence Model* definiert sich dadurch, dass es keine überlappenden Statements enthält und dass die Höhe der Statements entsprechend ihren Nachbarn angepasst wurden.
Um eine Überlappung aufzulösen, wird überprüft, welches der beiden Statements im überlappten Bereich das Stärkere ist. Das Schwächere der beiden wird dann in seiner Breite entsprechend reduziert. Dies ist in Abbildung 2.3b am Beispiel von Statement A und B zu sehen. Hier ist A im a-Intervall $[1.5, 2.5]$ schwächer als B und wurde entsprechend in seiner Breite reduziert. Für B und C gilt ebenfalls, dass B das stärkere Statement im überlappten Bereich ist und C wurde entsprechend in seiner Breite angepasst.
Die Anpassung in der Höhe ergibt sich aus dem Monotonieverhalten und der darauf definierten Ordnung sowie dem logischen Verlauf von Funktionen. Eine Höhenanpassung ist ebenfalls in Abbildung 2.3b zu sehen. Hier wurde Statement A in seiner Höhe reduziert. Dies lässt sich intuitiv damit erklären, dass Funktionen zwischen 0.5 und 1.5 mindestens konstant oder monoton steigend sein müssen und insbesondere nicht monoton fallend sein

können. Demnach können sie nicht über 2.5 steigen, da Statement B für den Bereich von 1.5 bis 4.5 vorgibt, dass Funktionen hier konstant verlaufen müssen, mit einem b-Wert zwischen 2 und 2.5. Funktionen, die vor 1.5 höher steigen würden, als 2.5, könnten aufgrund des definierten Monotonieverhaltens von Statements A, die Vorgaben von Statement C nicht mehr erfüllen. Diese Art von Höhenanpassung tritt immer auf, wenn zwei benachbarten Statements ein konstantes und steigendes bzw. fallendes Monotonieverhalten aufweisen. Um letztendlich ein stärkeres Statement als H zu erhalten, kann Statement B auf die Breite von H reduziert werden (Abbildung 2.3c). Dieses reduzierte Statement ist dann stärker als H, und somit ist gezeigt worden, dass sich H aus $\mathcal{M}$ ableiten lässt.

Dieses Verfahren kann genutzt werden, um zu überprüfen, ob eine Hypothese im Zusammenhang eines entsprechenden Experiments semantisch korrekt ist. Die Voraussetzung dafür ist, dass die Experimente als *Influence Model* vorliegen müssen und sich die Hypothesen in Statements übersetzt lassen. Im weiteren Verlauf der Arbeit wird untersucht, ob und wie diese Voraussetzungen erfüllt werden können.

3 Von Hypothesen zu Statements

Im vorherigen Kapitel wurden der *Calculus of Influence* und eine Grammatik für Hypothesen vorgestellt. Der Kalkül ermöglicht die Überprüfung der Semantik einer Hypothese, während die Grammatik die Syntax überprüft. In diesem Kapitel wird zuerst erarbeitet, wie Hypothesen in Statements überführt werden können. Anschließend wird erklärt, warum sich nicht alle Hypothesen übersetzen lassen. Abschließend wird untersucht, ob sich durch die Verknüpfung mit dem Kalkül neue Ableitungsregeln für die Grammatik ergeben und ob die Grammatik dadurch erweitert werden kann.

3.1 Eine Funktion zum Übersetzen von Hypothesen in Statements

In Kapitel 2.1 wurde bereits erklärt, wie die Durchführung eines Experiments abläuft. In diesem Kapitel bezieht sich der Begriff *Experiment* auf ein $\mathcal{V}$*-Influence Model* und eine Menge von Token, die passend zu einer Forschungsfrage gebildet wurden.

Sei also ein Experiment E gegeben, mit der Token-Menge T und dem Influence Model M. Dann existiert eine Menge H, die alle Hypothesen enthält, die auf Basis von T und der Grammatik des iLL gebildet werden können. Die Übersetzung von Hypothesen in Statements erfolgt dann durch die folgende Funktion:

$$\text{conv} : H \to M$$

Bei der Funktion conv (kurz für convert) handelt es sich um eine partielle Funktion, da nicht alle Hypothesen in Statements übersetzt werden können, wie im nächsten Kapitel erklärt wird. Die Abbildungsvorschrift wird basierend auf dem Aufbau der Grammatik definiert, da Hypothesen in ihrem Aufbau dieser unterliegen.
Um die Abbildungsvorschrift für conv definieren zu können, benötigt es noch die folgende Funktion.

Definition 3.1.1 *Sei* $V := \{(t, s, d, b, i) \mid t \in \{$ *UnspezifischeVariable, SpezifischeVariable* $\}$ *mit* s *beliebig und* $d = b = i = \_ \}$ *die Teilmenge von* T*, die Variablen repräsentieren für das Experiment* E*. Dann ist* $v : V \to \mathcal{V}$ *eine Funktion, die die Variablen des Experiments auf die Variablen des Influence Models* M *abbildet.*

In einem Experiment können verschiedene Variablen auftreten, die synonym zueinander sind. Beispielsweise können die Begriffe „Hefe“, „Hefeteig“ und „Pilz“ in einem Experiment vorkommen. Diese Begriffe sind Synonyme zueinander. Zur Abbildung dieser Variablen auf eine einzige Variable eines Influence Models wird die Funktion v verwendet. Zum Beispiel könnte $v((\text{UnspezifischeVariable}, \textit{Hefe}, \_, \_, \_)) = v((\text{UnspezifischeVariable}, \textit{Hefeteig}, \_, \_, \_)) = h$ sein.

Im Folgenden wird die Funktion conv anhand der einzelnen Ableitungsregeln der Grammatik des iLL definiert. Dabei werden die Symbole E, T, H und M gemäß den anfänglichen Definitionen im Kapitel verwendet. Des Weiteren sei $I_\infty = [-\infty, \infty]$ und Q die Menge von Hypothesen, die Token mit den Typen Quantität und SpezifischeQuantität enthalten.

Unspezifischer Satz: Sei $h \in H$ und es gilt *hypothese* $\Rightarrow$ *einfacherZusammenhang* $\Rightarrow$ *unspezifischerSatz* $\Rightarrow^*$ h. Sei $t_0t_1t_2$ die Tokenfolge, aus der h besteht, kurz $h = t_0t_1t_2$. Dann ist, $t_1 = (\text{UnspezifischeAbhängigkeit}, s, d, \rightsquigarrow, \_)$ wobei s beliebig und $d \in \{\triangleleft, \triangleright\}$ und conv wie folgt definiert:

$$\text{conv}(h) = (v(t_i), I_\infty, \rightsquigarrow, I_\infty, v(t_j)) \text{ mit } (i,j) := \begin{cases} (0,2) & \text{wenn } d \equiv \triangleright, \\ (2,0) & \text{sonst.} \end{cases}$$

Eine Hypothese, die über die Regel *einfacherZusammenhang* zu einem *unspezifischerSatz* abgeleitet wurde, besteht aus drei Token. Dabei repräsentieren der erste und der letzte Token jeweils eine unspezifische Variable, während der mittlere Token eine unspezifische Abhängigkeit darstellt.
Ein Token vom Typ UnspezifischeAbhängigkeit enthält neben dem Typ und dem Wortbaustein auch eine Richtung und ein Monotonieverhalten. Das Monotonieverhalten ist dabei immer $\rightsquigarrow$, da eine unspezifische Abhängigkeit lediglich beschreibt, dass es einen Zusammenhang zwischen den beiden Variablen gibt, ihn jedoch nicht genauer definiert. Aus dem gleichen Grund werden beide Intervalle auch größtmöglich gewählt. Die Richtung gibt an, welche Variable die abhängige und welche die unabhängige Variable ist. Wenn das Dreieck nach rechts zeigt, bedeutet dies, dass die erste Variable die unabhängige Variable ist und die zweite die abhängige. Zeigt das Dreieck in die entgegengesetzte Richtung, verhält es sich umgekehrt.
Die beiden Token, die die Variablen repräsentieren, werden mithilfe der Funktion v in die entsprechenden Variablen der Statements übersetzt.
Die Hypothese „Die Temperatur beeinflusst die Hefeaktivität“ mit den Token (UnspezifischeVariable, die Temperatur, _, _, _), (UnspezifischeAbhängigkeit, beeinflusst, $\triangleright, \rightsquigarrow, \_$), (UnspezifischeVariable, der Hefeteig, _, _, _) wird also zu dem Statement (a, $[-\infty, \infty]$, $\rightsquigarrow$, $[-\infty, \infty]$, b) übersetzt, wenn $v(t_0) = a$ und $v(t_2) = b$ ist.

Spezifischer Satz 1: Sei $h \in H \setminus Q$ und es gilt *hypothese* $\Rightarrow$ *einfacherZusammenhang* $\Rightarrow$ *spezifischerSatz* $\Rightarrow$ *abhängigeVariableSpezifischeAbhängikeit* $\Rightarrow^*$ h mit $h = t_0t_1t_2t_3$. Dann ist $t_1 = (\text{SpezifischeAbhängikeit}, s_1, d_1, b_1, \_)$ mit s_1, d_1 beliebig und $b_1 \in \{\nearrow, \searrow\}$

und $t_3 = (k, s_3, d_3, b_3, i)$ mit s_3, d_3, b_3 beliebig, i ein Intervall über $\mathbb{R}_\infty$ oder _ und $k \in \{\text{UnspezifischeVariable}, \text{SpezifischeVariable}\}$ und conv wie folgt definiert:

$$\text{conv}(h) = (v(t_3), I, b, I_\infty, v(t_0)) \text{ mit } I := \begin{cases} i & \text{wenn } k = \text{SpezifischeVariable}, \\ I_\infty & \text{sonst.} \end{cases}$$

Eine Hypothese, die durch die Ableitungsregeln von *einfacherZusammenhang* über *spezifischerSatz* und *abhängigeVariableSpezifischeAbhängigkeit* entsteht und keine Token, mit den Typen Quantität, SpezifischeQuantität enthält, besteht aus vier Token. Das erste Token ist eine UnspezifischeVariable. Es entspricht in dieser Ableitungsregel immer der abhängigen Variable, wodurch die Positionierung im Statement durch die Ableitungsregel festgelegt ist.
Das vierte Token entspricht demnach der unabhängigen Variable. Es kann entweder eine UnspezifischeVariable oder eine SpezifischeVariable sein. Wenn es sich um eine SpezifischeVariable handelt, wird neben der ersten Variable des Statements auch das Intervall des Tokens für das erste Intervall im Statement verwendet. Ein Beispiel für ein solches Token ist (SpezifischeVariable, 20 °C, _, _, [20, 20]).
Das zweite Token ist eine SpezifischeAbhängigkeit, die hier eine Aussage über das Verhalten der abhängigen Variable trifft. Spezifische Abhängigkeiten dienen hier dazu, ein steigendes oder fallendes Verhalten der abhängigen Variable zu beschreiben. Das Token enthält daher eins der beiden Monotonieverhalten $\nearrow$ oder $\searrow$, das direkt in das Statement übertragen wird. Ein Beispiel für ein solches Token ist (SpezifischeAbhängigkeit, steigt, _, $\nearrow$, _).
Das dritte Token ist (Bei, bei, _, _, _). Es ist ein Hilfstoken, das dazu dient, Sätze zu formulieren, die auch aus deutsch-grammatikalischer Sicht korrekt sind, jedoch keine mathematische Bedeutung haben. Ein Beispiel für solch eine Hypothese in Form von Token ist:
(UnspezifischeVariable, Der Pilz, _, _, _), (SpezifischeAbhängigkeit, wächst, _, $\nearrow$, _), (Bei, bei, _, _, _), (SpezifischeVariable, 20 °C, _, _, [20, 20]]).

Spezifischer Satz 2: Sei $h \in H \setminus Q$ und es gilt *hypothese* $\Rightarrow$ *einfacherZusammenhang* $\Rightarrow$ *spezifischerSatz* $\Rightarrow$ *spezifischeAbhängigkeitAbhängigeVariable* $\Rightarrow^*$ h mit $h = t_0 t_1 t_2 t_3$. Dann ist $t_2 = (\text{SpezifischeAbhängigkeit}, s_2, d_2, b_2, \_)$ mit s_2, d_2 beliebig und $b_1 \in \{\nearrow, \searrow\}$ und $t_1 = (k, s_1, d_1, b_1, i)$ mit s_1, d_1, b_1 beliebig, i ein Intervall über $\mathbb{R}_\infty$ oder _ und $k \in \{\text{UnspezifischeVariable}, \text{SpezifischeVariable}\}$ und conv wie folgt definiert:

$$\text{conv}(h) = (v(t_1), I, b, I_\infty, v(t_3)) \text{ mit } I := \begin{cases} i & \text{wenn } k \equiv \text{SpezifischeVariable}, \\ I_\infty & \text{sonst.} \end{cases}$$

Eine Hypothese, die über *einfacherZusammenhang*, *spezifischerSatz* zu einem *spezifischeAbhängigkeitAbhängigeVariable* abgeleitet wurde und keine Token mit den Typen Quantität, SpezifischeQuantität enthält, besteht ebenfalls aus vier Token, für die fast das Gleiche gilt wie für den vorherigen Fall. Der Unterschied liegt in der vertauschten Reihenfolge der Variablen und der Positionierung des Hilfstoken *Bei* am Satzanfang. Die unabhängige Variable steht in dieser Satzform an zweiter Stelle und die abhängige Variable entsprechend am Satzende. Der Rest ist identisch zum **Spezifischen Satz 1**.
Ein Beispiel für solch eine Hypothese in Form von Token ist:
(Bei, Bei, _, _, _), (SpezifischeVariable, 0 °C, _, _, [0, 0]), (SpezifischeAbhängigkeit, fällt, _, $\searrow$, _), (UnspezifischeVariable, der Hefeteig, _, _, _).

Konditional Satz: Sei $h \in H \setminus Q$ und es gilt *hypothese* $\Rightarrow$ *einfacherZusammenhang* $\Rightarrow$ *konditionalsatz* $\Rightarrow^*$ h mit $h = t_0t_1t_2t_3t_4t_5$. Dann ist $t_2 =$ (SpezifischeAbhängigkeit, s_2, _, b_2, _) und $t_4 =$ (SpezifischeAbhängigkeit, s_4, _, b_4, _) mit $b_2, b_4 \in \{\nearrow, \searrow\}$ und s_2, s_4 beliebig und conv wie folgt definiert:

$$\text{conv}(h) = (v(t_1), I_\infty, b, I_\infty, v(t_5)) \text{ mit } b := \begin{cases} b_4 & \text{wenn } b_2 = \nearrow, \\ \text{invert}(b_4) & \text{sonst.} \end{cases}$$

Eine Hypothese, die über den *einfacherZusammenhang* zu einem *konditionalSatz* abgeleitet wurde und keine Token mit den Typen Quantität, SpezifischeQuantität enthält, besteht aus sechs Token. Der erste und der vierte Token bilden das Konditional-paar, das immer gemeinsam auftritt. Beispiele hierfür sind „Wenn“ und „, dann“ oder „Je“ und „, desto“. Ein Token für den Konditional enthält keine weiteren Informationen außer dem Typ und dem Wortbaustein. Der zweite und der letzte Token repräsentieren jeweils die Variablen. In dieser Satzform ist der zweite Token immer die unabhängige Variable, während der letzte Token die abhängige Variable darstellt und somit die Platzierung im Statement klar ist. Der dritte und der fünfte Token sind jeweils vom Typ *SpezifischeAbhängigkeit* und beschreiben das Verhalten der benachbarten Variable. Das Verhalten des Statements ergibt sich aus der Kombination dieser beiden Verhalten. Wenn das Verhalten der unabhängigen Variable $\nearrow$ ist, kann das Verhalten der abhängigen Variable für das Statement übernommen werden. Wenn das Verhalten der unabhängigen Variable jedoch $\searrow$ ist, muss das Verhalten der abhängigen Variable invertiert werden (= invert-Funktion). Das bedeutet, dass $\nearrow$ zu $\searrow$ wird und umgekehrt. Wie bereits erklärt, kann der Zusammenhang zwischen zwei Variablen mit einer Funktion in einem Koordinatensystem dargestellt werden, wobei die horizontale Achse die Werte der unabhängigen Variable repräsentiert und die vertikale Achse die Werte der abhängigen Variable. Demnach kann ein $\searrow$-Verhalten der unabhängigen Variable als die Leserichtung des Graphen von rechts nach links interpretiert werden, woraus sich ein invertiertes Verhalten der abhängigen Variable für das Statement ergibt, weil dieses auf der herkömmlichen Betrachtungsweise eines Graphen beruht. Ein Beispiel für solch eine Hypothese in Form von Token ist:
(Konditional, Wenn, _, _, _), (UnspezifischeVariable, die Temperatur, _, _, _),
(SpezifischeAbhängigkeit, steigt, _, $\nearrow$, _), (Konditional, , dann, _, _, _),
(SpezifischeAbhängigkeit, steigt, _, $\nearrow$, _), (UnspezifischeVariable, der Hefeteig, _, _, _).

Einschränkung Sei $h \in H \setminus Q$ und es gilt *hypothese* $\Rightarrow$ *komplexerZusammenhang* $\Rightarrow^*$ h mit $h = t_0...t_n$, wobei *einfacherZusammenhang* $\Rightarrow^*$ h_0 mit $h_0 = t_0...t_{n-2}$ und *einschränkung* $\Rightarrow^*$ h_1 mit $h_1 = t_{n-1}t_n$. Dann ist $t_n =$ (Wertebereich, s, _, _, I) mit s beliebig und I ein $R_\infty = \mathbb{R} \cup \{-\infty, \infty\}$ Intervall und conv wie folgt definiert:

$$\text{conv}(h) = \text{conv}'(\text{conv}(h_0)) = (v_0, I, b, I_\infty, v_1) \text{ mit } v_0, v_1 \in \mathcal{V}, b \in \{\rightarrow, \nearrow, \searrow \rightsquigarrow\},$$

wobei conv′ das erste Intervall eines Statements durch das der Einschränkung ersetzt.

Eine Hypothese, die über die Regel *komplexerZusammenhang* abgeleitet wird und kein *vergleichSatz* ist, besteht aus zwei Teilen. Der erste Teil ist eine Satzform des *einfacher-Zusammenhang* und der zweite Teil ist eine *einschränkung*. Um eine Hypothese dieser

Form in ein Statement zu überführen, wird der erste Teil wie zuvor beschrieben mit der conv-Funktion in ein Statement übersetzt. Auf das resultierende Statement wird dann die conv′-Funktion angewandt, die das erste Intervall, durch das der Einschränkung ersetzt. Ein Beispiel für solch eine Hypothese in Form von Token ist:
(UnspezifischeVariable, der Hefeteig, _, _, _) (UnspezifischeAbhängigkeit, wird beeinflusst von, $\lhd$, $\rightsquigarrow$, _) (UnspezifischeVariable, der Temperatur, _, _, _) (Konnektor, aber nur, _, _, _) (Wertebereich, von 10 °C bis 50 °C, _, _, $[10, 50]$).

3.2 Grenzen der Übersetzbarkeit

Die zuvor definierte conv-Funktion deckt nicht alle Satzformen der Grammatik ab und wurde teilweise mit Einschränkungen definiert. Dies liegt daran, dass sich unter den derzeitigen Gegebenheiten nicht alle Satzformen in Statements übersetzen lassen oder nur unter bestimmten Einschränkungen. In diesem Kapitel wird erarbeitet, warum dies so ist.

Wie sich bereits aus der Definition der conv-Funktion ablesen lässt, können Hypothesen, die Token der Typen Quantität und Temporal enthalten, nicht in ein Statement übersetzt werden. Dies betrifft einige Varianten der Ableitungsregeln *spezifischerSatz*, *konditionalsatz* und *vergleichSatz*. Darüber hinaus lässt sich keine der Satzformen von *vergleichSatz* übersetzen.

Temporal. Temporal-Token beziehen sich immer auf eine spezifische Abhängigkeit, und eine spezifische Abhängigkeit beschreibt eine Veränderung einer Variable. Ein Temporal ermöglicht es, eine spezifische Abhängigkeit, um eine zeitliche Komponente zu erweitern, wodurch Formulierungen wie „Der Hefeteig steigt schneller“ möglich werden, wobei „schneller“ das Temporal ist. Ein Beispiel für eine Hypothese mit einem Temporal wäre „Bei 40°C steigt der Hefeteig am schnellsten“.
Bei der Übersetzung von Hypothesen in Statements wurden den Token und den dazugehörigen Wörtern mathematische Eigenschaften zugeschrieben. So entspricht eine spezifische Abhängigkeit einem Verhalten, nämlich $\nearrow$ oder $\searrow$, um anzuzeigen, ob die Werte einer Variable zunehmen oder abnehmen. Das Monotonieverhalten im Statement wird daraus abgeleitet. Ein Temporal trifft letztendlich also eine Aussage darüber, wie schnell eine Funktion steigt oder fällt. Mathematisch betrachtet entspricht dies einer Aussage über die erste Ableitung einer Funktion, die auch noch von der Zeit abhängt. Konkret entspricht dies einer zweistelligen Funktion der Form $f(t, \text{Temperatur}) = \text{Hefeteig}$, wobei t die Zeit darstellt.
Da ein Statement eine Abstraktion eines Funktionsausschnittes von einstelligen Funktionen darstellt, bei dem keine Informationen über Steigung enthalten sind, können Temporal und Hypothesen, die diese enthalten, nicht direkt in ein Statement übersetzt werden.

Quantität und Vergleiche. Ein Quantitätstoken erscheint in den Ableitungsregeln *spezifischerSatz*, *konditionalSatz* und *vergleichSatz*.
In den Satzformen der Ableitungsregeln *spezifischerSatz* und *vergleichSatz* bezieht es sich

auf eine unspezifische Variable, was Formulierungen wie „mehr Licht“ ermöglicht, wobei „mehr“ die Quantität ist.
In den Satzformen der *konditionalSatz*-Regel bezieht sich die Quantität auf die spezifischen Abhängigkeiten und ermöglicht Formulierungen wie „Je weniger die Temperatur steigt“, wobei „weniger“ die Quantität und „steigt“ die spezifische Abhängigkeit ist.

Wenn sich die Quantität auf eine spezifische Abhängigkeit bezieht, beschreibt sie den Grad des Monotonieverhaltens, also ob es stärker oder schwächer ist. Dies entspricht einer Aussage über die erste Ableitung einer Funktion, die ebenfalls von der Zeit abhängt. Dies lässt sich aus analogen Gründen wie bei den Temporalen nicht übersetzen. Dies erklärt, warum die Satzformen der *konditionalSatz*-Regel nicht übersetzt werden können. Hypothesen dieser Satzformen sind zum Beispiel „Wenn die Temperatur stärker steigt, dann steigt die Hefeaktivität stärker“ oder „Je geringer die Temperatur fällt, desto geringer steigt der Hefeteig“.

Das Paper „The Calculus of Temporal Influence“ [2] diskutiert Ansätze, wie sich auch die Zeit modellieren lässt, aber die Erkenntnisse sind noch nicht in diese Arbeit eingeflossen, da dies größere Änderungen beim Anlegen der Experimente mit sich bringen würde.

In den Satzformen der *spezifischerSatz*-Regel, die Quantitäten enthalten. Hier bezieht sich die Quantität auf die unabhängige Variable. Beispiele für Hypothesen dieser Satzformen sind „Der Hefeteig steigt bei höher Temperatur“ und „Bei geringer Temperatur fällt der Hefeteig“ mit den Quantitäten „höher“ und „geringer“. Diese Hypothesen sind zu ungenau formuliert, um in Statements übersetzt zu werden. Es ist nicht klar, was als „höhere“ oder „geringere“ Temperatur gilt, und daher kann dies nicht in ein Statement überführt werden.
Dass solche unpräzisen Hypothesen von der Grammatik unterstützt werden, liegt in der Lernchance, die sie für Lernende bieten. Wenn versucht wird, eine solche Hypothese in einem Experiment zu überprüfen, sollten die Lernenden erkennen, dass sie ungenau formuliert ist.

Bei den Satzformen der *vergleichSatz*-Regel liegt ein Problem darin, dass ein Vergleich darüber angestellt wird, wie zwei verschiedene Ausprägungen der unabhängigen Variable den Beeinflussungsgrad auf die abhängige Variable verändern. Zwei Beispiele dafür sind: „Der Hefeteig steigt bei 40°C besser als bei 10°C“ und „Höhere Temperatur beeinflusst den Hefeteig besser als geringere Temperatur“. Das bedeutet, solch eine Hypothese müsste in mehr als ein Statement übersetzt werden, da ein Vergleich im Allgemeinen zwischen mindestens zwei Entitäten gezogen wird. Aber selbst wenn die Hypothese in zwei oder mehr Statements übersetzt werden würde, bezieht sich der Vergleich wieder auf die Steigung der zugrunde liegenden Funktion, da gefragt wird, welche Ausprägung der unabhängigen Variable eine bessere/schlechtere Auswirkung auf die abhängige Variable hat, und diese lässt sich nicht übersetzen.

3.3 Erweiterungsmöglichkeiten

In diesem Kapitel werden zwei Aspekte genauer untersucht. Als Erstes wird analysiert, ob durch die Auswertung der Statements weitere Hypothesen generiert werden können. Des

Weiteren wird untersucht, ob es möglich ist, komplexere Fragen an ein Influence Model zu stellen, die über die bisherigen Möglichkeiten des Calculus of Influence hinausgehen.

Analyse der Statements

Für die Analyse werden nur Statements mit den Variablen X und Y betrachtet, wobei X die unabhängige Variable und Y die abhängige Variable ist. Des Weiteren wird zwischen offenen, halboffenen, geschlossenen und Punktintervallen unterschieden. Das Monotonieverhalten wird wie gewohnt in die vier bekannten Verhaltensweisen unterteilt. Aus dieser Betrachtung ergeben sich insgesamt 100 unterschiedliche Statements.
Bei der Untersuchung dieser 100 Aussagen lassen sich drei Kategorien von Satzformen identifizieren, die auf der Ausprägung der Intervalle basieren. Es ist anzumerken, dass die folgenden Satzformen sehr rudimentär und mathematisch formuliert sind. Sie dienen lediglich als Vorschlag und Muster, um daraus Grammatikregeln abzuleiten, mit denen sich Hypothesen formulieren lassen, die auch der deutschen Sprache gerecht werden.
Des Weiteren gilt im Folgenden, dass X, Y Variablen sind, $b \in \{\rightarrow, \nearrow, \searrow, \rightsquigarrow\}$, $I_\infty = [-\infty, \infty]$ und $x_0, y_0, x_l, x_u, y_l, y_u \in \mathbb{R}$.

Offene Intervalle. Statements der Form $(X, I_\infty, b, I_\infty, Y)$ dienen als Übersetzungsziel für Hypothesen, die allgemeine Aussagen über den Zusammenhang der beiden Variablen treffen. Sie treffen nur Aussagen über das Monotonieverhalten. Solche Hypothesen haben die Form „X beeinflusst Y“ oder „Wenn X steigt/fällt, dann steigt Y/fällt Y/ist Y konstant“. Hypothesen dieser Art werden zum größten Teil bereits von der Grammatik abgedeckt. Hier liegt eine mögliche Erweiterung in Hypothesen, die behaupten, dass die abhängige Variable über den ganzen Definitionsbereich konstant ist. Beispielsweise könnte eine solche Hypothese „Wenn die Temperatur steigt verändert sich der Hefeteig nicht“ lauten.

Halboffene und geschlossene Intervalle. Statements der Formen $(X, I_\infty, b, [y_l, y_u], Y)$, $(X, [x_l, x_u], b, I_\infty, Y)$ und $(X, [x_l, x_u], b, [y_l, y_u], Y)$ dienen als Übersetzungsziel für Hypothesen, deren Aussage nur über einen Teil des Definitionsbereichs spricht oder die obere und untere Schranken für den Wertebereich enthalten oder eine Kombination der beiden Varianten. Formen dieser Hypothesen sind die bereits in der Grammatik enthaltenen „X beeinflusst Y ab x_l/bis x_u/von x_l bis x_u“. Noch nicht enthaltenen Formen sind „X beeinflusst Y und Y wird nicht größer als y_u/kleiner als y_l/bleibt zwischen y_l und y_u“ und „Wenn X ab x_l/bis x_u/zwischen x_l und x_u steigt/fällt, dann steigt/fällt Y bis höchstens y_u/y_l/zwischen y_u und y_l“. Für das Temperatur-Hefe-Beispiel könnten die Hypothesen „Die Temperatur beeinflusst den Hefeteig bis 43°C und der Hefeteig wird nicht höher als 41 cm“ oder „Wenn die Temperatur zwischen 2°C und 37°C steigt, dann steigt der Hefeteig zwischen 5 cm und 19 cm“.

Punktintervall. Statements der Form $(X, [x_0, x_0], b, I, Y)$, $(X, I, b, [y_0, y_0], Y)$ und $(X, [x_0, x_0], b, [y_0, y_0], Y)$, wobei I ein beliebiges Intervall über $\mathbb{R}_\infty$ ist, dienen als Übersetzungsziel für Hypothesen, die Aussagen darüber treffen, wie sich die abhängige Variable

bei konkreten, Werten der unabhängigen Variable verhält oder die besagen, dass die abhängige Variable unter bestimmten Voraussetzungen einen spezifischen Wert annimmt. Um letzteres sinnvoll nutzen zu können, muss beim Anlegen eines Influence Models darauf geachtet werden, dass gezielt Statements mit der Höhe $[0, 0]$ angelegt werden. Andernfalls würden Überprüfungen für Hypothesen dieser Form immer negativ ausfallen. Die Formen solcher Hypothesen sind „x_0 beeinflusst Y ...“, „Bei x_0 steigt/fällt Y ...“, „... und Y hat den Wert y_0“ und „Bei x_0 nimmt Y den Wert y_0 an“. Beispiele für Hypothesen könnten lauten „23°C beeinflussen den Hefeteig und der Hefeteig hat eine höhe von 17 cm“ und „Bei 42°C ist der Hefeteig 53 cm hoch“

Normalisiertes Influence Model

Bisher wurde das Calculus of Influence genutzt, um zu überprüfen, ob sich eine Hypothese in Form eines Statements von einem Influence Model ableiten lässt. Das Kalkül kann jedoch auch genutzt werden, um aus einem beliebigen Influence Model M ein normalisiertes Influence Model M' zu erzeugen (siehe 2.3). Für die Statements in M' gilt, dass sich für sie keine stärkeren Statements mehr finden lassen. An ein solches Modell M' können verschiedene Fragen gestellt und beantwortet werden, die sich auch als Hypothesen für ein Experiment eignen.

Sei im Folgenden N ein normalisiertes $\{X, Y\}$-Influence Model, $b, b_i \in \{\rightarrow, \nearrow, \searrow, \rightsquigarrow\}$ ein beliebiges Verhalten, alle $x_j, y_k \in \mathbb{R}_\infty$ mit $i, j, k \in \{0, 1, 2, ..., n\}$.

Des Weiteren wird die Menge der Statements um eine weitere Form von Statement erweitert. Ein Statement $S = (t, v_1, v_2, x, Y)$ wird eingeführt, wobei $t \in \{\mathsf{vergleich}, \mathsf{max}, \mathsf{min}\}$ den Typ angibt und v_1 ein Intervall oder ein Wert aus $\mathbb{R}_\infty$ ist, während v_2 immer ein Intervall aus $\mathbb{R}_\infty$ ist. Die Variablen x und Y können weggelassen werden, sofern sie aus dem Kontext klar ersichtlich sind.

Vergleich der Funktionswerte. Ein Vergleich der Funktionswerte entspricht einer Hypothese der Form „Im Intervall $[x_{l_g}, x_{r_g}]$ ist Y größer als im Intervall $[x_{l_s}, x_{r_s}]$“. Hypothesen dieser Form können als Statement $H = (\mathsf{vergleich}, [x_{l_g}, x_{r_g}], [x_{l_s}, x_{r_s}])$ übersetzt werden, wobei vergleich den Typ angibt.
Angenommen, y_{b_g} ist die kleinste untere Grenze des Y-Intervalls für alle Statements im Intervall $[x_{l_g}, x_{r_g}]$, und y_{t_s} ist die größte obere Grenze des Y-Intervalls für alle Statements im Intervall $[x_{l_s}, x_{r_s}]$. Eine vergleich-Hypothese ist genau dann korrekt, wenn $y_{t_s} \leq y_{b_g}$ gilt.

Abbildung 3.1 zeigt das Modell $M = \{A = (X, [0, 2], \nearrow, [0, 2], Y), B = (X, [2, 4], \nearrow, [1, 4], Y), C = (X, [4, 6], \searrow, [3, 5], Y)\}$. In diesem Modell ist die Hypothese $H_1 = (\mathsf{vergleich}, [5, 5], [1, 1])$ korrekt, da die untere Kante von C mit einem Y-Wert von drei höher liegt als die obere Kante von A mit einem Y-Wert von zwei. Für die Hypothese $H_2 = (\mathsf{vergleich}, [3, 3], [1, 1])$ ist dies nicht der Fall, da die untere Kante von B mit einem Y-Wert von eins niedriger liegt als die obere Kante von A.
Ein Beispiel für eine Hypothese dieser Art könnte lauten: "Bei 41 °C ist die Hefeaktivität größer als bei 23 °C."

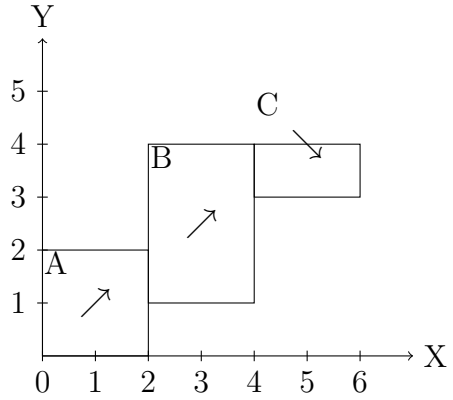


Abbildung 3.1: Influence Model M bei dem C größer A und B nicht größer A ist.

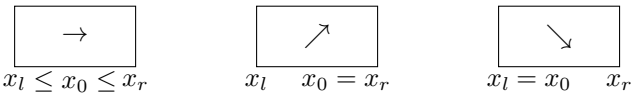


Abbildung 3.2: Die drei Fälle für ein lokales Maximum, wenn das Intervall $[x_l, x_r]$ höchstens ein Statement umfassen

Lokale Extremstellen. Eine Hypothese, die eine Aussage bezüglich eines Extremums macht, hat die Form „Im Intervall $[x_l, x_r]$ gibt es bei x_0 ein lokales Maximum“, die als $H_2 = (\mathsf{max}, x_0, [x_l, x_r])$ mit $x_l \leq x_0 \leq x_r$ übersetzt wird. Es gibt verschiedene Fälle, in denen eine solche Hypothese als korrekt gilt.

Für den Spezialfall, dass $x_l = x_0 = x_r$ gilt, ist die Hypothese immer korrekt.

Des Weiteren gibt es den Fall, dass das Intervall $[x_l, x_r]$ höchstens ein Statement $S = (X, [x_1, x_2], b, I_\infty, Y)$ mit $x_1 \leq x_l < x_r \leq x_2$ umfasst. In diesem Fall ist die Hypothese genau dann korrekt, wenn entweder $b = \rightarrow$ oder $b = \nearrow$ gilt und $x_0 = x_r$ oder $b = \searrow$ gilt und $x_0 = x_l$ (siehe Abbildung 3.2).

Zuletzt gibt es noch mehrere Fälle, bei denen das Intervall $[x_l, x_r]$ mehr als ein Statement in N umfasst.

Fall 1:
Die Hypothese ist genau dann korrekt, wenn die benachbarten Statements $S = (X, [x_1, x_2], \searrow, [y_B, y_T], Y)$, $(X, [x_2, x_3], b_1, [y_{b_1}, y_{t_2}], Y)$, ..., $(X, [x_{n-1}, x_n], b_n, [y_{b_n}, y_{t_n}], Y)$ mit $x_1 \leq x_l$, $x_n \leq x_r$, $b_1, ..., b_n \in \{\searrow, \rightarrow\}$ und es gilt das $x_0 = x_l$ ist. Des Weiteren muss für alle anderen Statements $(X, I_\infty, b, [y_b, y_t], Y)$ in $[x_l, x_r]$ gelten, dass $y_t \leq y_B$.

Dieser Fall beschreibt, dass sich das lokale Maximum am linken Rand des Intervalls befindet, repräsentiert durch das Statement S, das ein fallendes Monotonieverhalten aufweist. Es ist auf der rechten Seite direkt von beliebig vielen Statements umgeben, die ebenfalls ein fallendes oder konstantes Monotonieverhalten haben. Sofern es noch weitere Statements in diesem Intervall gibt, müssen deren obere Kanten unterhalb der unteren Kante von S liegen. Wenn alle diese Voraussetzungen erfüllt sind und der Wert des linken Intervall-Randes als Wert für das lokale Maximum angegeben wurde ($x_0 = x_l$), ist die Hypothese korrekt. In Abbildung 3.3 ist im linken Diagramm eine Ausprägung dieses Falles dargestellt.

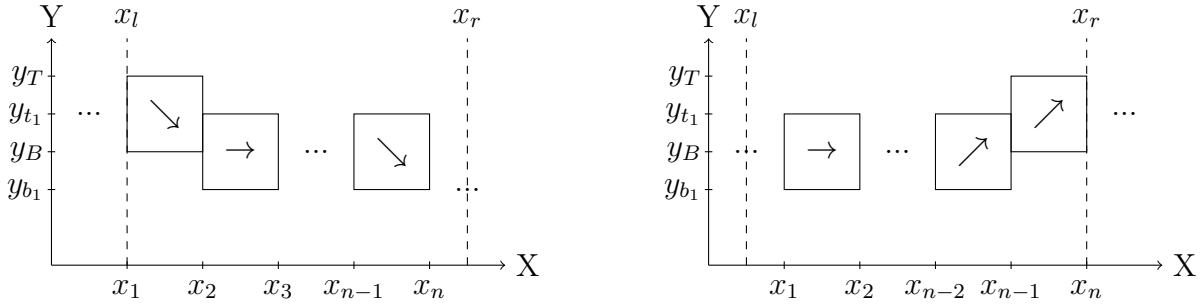


Abbildung 3.3: Zwei lokale Maxima, die sich am linken (links) und rechten (rechts) Rand des Intervalls befinden.

Fall 2:
Die Hypothese ist genau dann korrekt, wenn die benachbarten Statements $(X, [x_1, x_2], b_1, [y_{b_1}, y_{t_1}], Y)$, ..., $(X, [x_{n-2}, x_{n-1}], b_n, [y_{b_n}, y_{t_n}], Y)$, $S = (X, [x_{n-1}, x_n], \nearrow, [y_B, y_T], Y)$ mit $x_l \leq x_1$, $x_r \leq x_n$, $b_1, ..., b_n \in \{\nearrow, \rightarrow\}$ gibt und es gilt das $x_0 = x_r$ ist. Des Weiteren muss für alle anderen Statements $(X, I_\infty, b, [y_b, y_t], Y)$ in $[x_l, x_r]$ gelten, dass $y_t \leq y_B$.

Dieser Fall beschreibt, dass sich das lokale Maximum am rechten Rand des Intervalls befindet, repräsentiert durch das Statement S, das ein steigendes Monotonieverhalten aufweist. Es ist auf der linken Seite von beliebig vielen Statements umgeben, die ebenfalls ein steigendes oder konstantes Monotonieverhalten haben. Sofern es noch weitere Statements in diesem Intervall gibt, müssen deren obere Kanten unterhalb der unteren Kante von S liegen. Wenn alle diese Voraussetzungen erfüllt sind und der Wert des rechten Intervall-Randes als Wert für das lokale Maximum angegeben wurde ($x_0 = x_r$), ist die Hypothese korrekt. In Abbildung 3.3 ist im rechten Diagramm eine Ausprägung dieses Falles dargestellt.

Fall 3:
Die Hypothese ist genau dann korrekt, wenn es die benachbarten Statements $(X, [x_1, x_2], \nearrow, [y_1, y_T], Y)$, $S_1 = (X, [x_2, x_3], \rightarrow, [y_B, y_T], Y)$,..., $S_n = (X, [x_{n-1}, x_n], \rightarrow, [y_B, y_T], Y)$ mit $x_l < x_1$, $x_r \leq x_n$ gibt und $x_2 \leq x_0 \leq x_r$ gilt. Des Weiteren muss für alle anderen Statements $(X, I_\infty, b, [y_b, y_t], Y)$ in $[x_l, x_r]$ gelten, dass $y_t \leq y_B$.

Dieser Fall beschreibt, dass sich das lokale Maximum am Rand befindet, diesmal jedoch in Form eines oder mehrerer Statements mit konstantem Monotonieverhalten, die auf der rechten Seite von einem Statement mit steigendem Monotonieverhalten abgeschlossen werden. Sofern es weitere Statements in dem Intervall gibt, müssen deren obere Kanten unterhalb der Statements $S_1, ..., S_n$ liegen. Wenn all dies zutrifft und für x_0 ein Wert zwischen x_r und x_2 gewählt wurde, ist die Hypothese korrekt. In Abbildung 3.4 ist im linken Diagramm eine Ausprägung dieses Falles dargestellt.

Fall 4:
Die Hypothese ist genau dann korrekt, wenn es die benachbarten Statements $(X, [x_1, x_2], \rightarrow, [y_B, y_T], Y)$, ..., $(X, [x_{n-2}, x_{n-1}], \rightarrow, [y_B, y_T], Y)$, $(X, [x_{n-1}, x_n], \searrow, [y_1, y_T], Y)$ mit $x_1 \leq x_l$, $x_n < x_r$ gibt und $x_2 \leq x_0 \leq x_r$ gilt. Des Weiteren muss für alle anderen Statements $(X, I_\infty, b, [y_b, y_t], Y)$ in $[x_l, x_r]$ gelten, dass $y_t \leq y_B$.

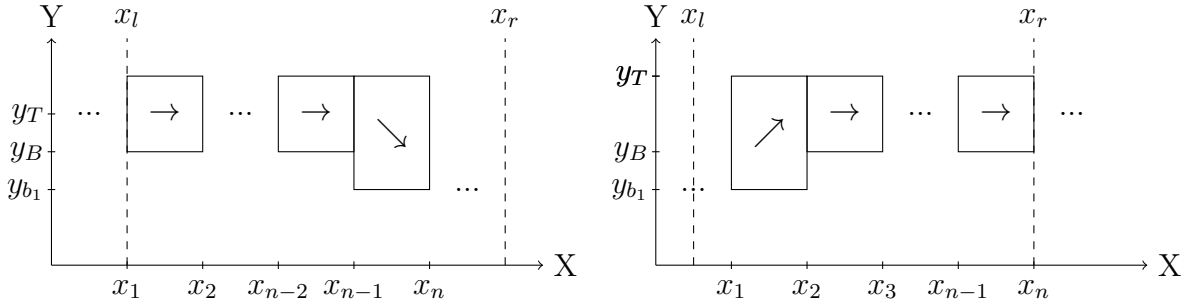


Abbildung 3.4: Zwei lokale Maxima, die sich am linken bzw. rechten Rand befinden.

Hier befindet sich das lokale Maximum am linken Rand, und alle weiteren Bedingungen sind analog zu Fall 3, jedoch vertikal gespiegelt. In Abbildung 3.4 ist im rechten Diagramm eine Ausprägung dieses Falles dargestellt.

Fall 5:
Die Hypothese ist genau dann korrekt, wenn es die benachbarten Statements $(X, [x_1, x_2], \nearrow, [y_1, y_T], Y)$, $S_1 = (X, [x_2, x_3], \rightarrow, [y_B, y_T], Y)$, ..., $S_2 = (X, [x_{n-2}, x_{n-1}], \rightarrow, [y_B, y_T], Y)$, $(X, [x_{n-1}, x_n], \searrow, [y_2, y_T], Y)$ mit $x_l < x_1$, $x_n < x_r$ gibt und $x_2 \leq x_0 \leq x_{n-1}$ gilt. Des Weiteren muss für alle anderen Statements $(X, I_\infty, b, [y_b, y_t], Y)$ in $[x_l, x_r]$ gelten, dass $y_t \leq y_B$.

Dieser Fall beschreibt, dass sich das lokale Maximum im Intervall befindet und nicht am Rand. Hier müssen ein oder mehrere Statements $S_1, ..., S_n$ mit konstantem Monotonieverhalten, auf der linken Seite von Statements mit steigendem Monotonieverhalten und auf der rechten Seite von Statements mit fallendem Monotonieverhalten abgeschlossen werden. Zusätzlich müssen $S_1, ..., S_n$ höher liegen als alle weiteren Statements im Intervall, sofern diese vorhanden sind. Wenn all diese Bedingungen erfüllt sind und x_0 so gewählt wurde, dass es sich im Intervall von x_2 bis x_{n-1} befindet, ist die Hypothese korrekt. Abbildung 3.5 veranschaulicht dies.

Fall 6:
Die Hypothese ist genau dann korrekt, wenn es die Statements $(X, [x_1, x_2], b_1, [y_{b_1}, y_{t_1}], Y)$, ... $(X, [x_{m-2}, x_{m-1}], b_m, [y_{b_2}, y_{t_m}], Y)$, $S_1 = (X, [x_m, x_0], \nearrow, [y_{B_1}, y_T], Y)$, $S_2 = (X, [x_0, x_{m+1}], \searrow, [y_{B_2}, y_T], Y)$, ... $(X, [x_{n-2}, x_{n-1}], b_{m+1}[y_{b_{n-1}}, y_{t_{n-1}}], Y)$, ... $(X, [x_{n-1}, x_n], b_n[y_{b_n}, y_{t_n}], Y)$ mit $x_l \leq x_1$, $x_n \leq x_r$, $b_1, ..., b_m \in \{\nearrow, \rightarrow\}$, $b_{m+1}, ..., b_n \in \{\searrow, \rightarrow\}$ gibt. O.B.d.A sei $y_{B_1} \leq y_{B_2}$, dann muss für alle anderen Statements $(X, I_\infty, b, [y_b, y_t], Y)$ in $[x_l, x_r]$ gelten, dass $y_t \leq y_{B_1}$.

Hier befindet sich das lokale Maximum erneut innerhalb des Intervalls und nicht am Rand. Es wird durch zwei Statements, S_1 und S_2, gebildet, die auf der linken Seite von Statements mit steigendem oder konstantem Verhalten und auf der rechten Seite von Statements mit fallendem oder konstantem Verhalten umgeben sind. Sofern es weitere Statements im Intervall gibt, müssen diese unterhalb von S_1 und S_2 liegen. Wenn all diese Bedingungen erfüllt sind und x_0 so gewählt wurde, dass es genau in der Mitte von S_1 und S_2 liegt, ist die Hypothese korrekt. Abbildung 3.5 veranschaulicht dies.

Dies sind alle Fälle, mit denen eine Hypothese für ein lokales Maximum überprüft werden kann, analog gilt dies auch für ein lokales Minimum. Es ist jedoch wichtig zu beachten,

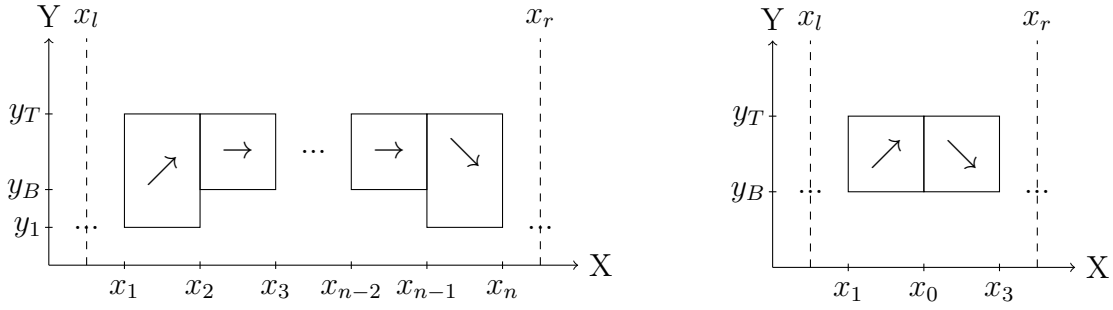


Abbildung 3.5: Zwei lokale Maxima, die sich nicht am Rand befinden.

dass nicht alle möglichen lokalen Extrema abgedeckt werden können, da dies von der Definition eines Statements abhängt, das letztendlich eine Funktionsschar beschreibt.
Es gibt Situationen, in denen sich zwei Kandidaten, K_1 und K_2, innerhalb eines Intervalls befinden. Ein Kandidat besteht aus einer Liste von Statements, die einem der beschriebenen Fälle entsprechen. Wenn die obere Kante von K_1 zwischen der oberen und unteren Kante von K_2 liegt, kann eine Hypothese, die ein lokales Extremum im Bereich eines der beiden Kandidaten angibt, nicht eindeutig überprüft werden. Es lässt sich immer ein Gegenbeispiel finden, bei dem sich das Extremum im anderen Kandidaten befindet.

Zusammenfassend kann festgestellt werden, dass es aus theoretischer Sicht verschiedene Möglichkeiten gibt, die Menge der Hypothesen für das iLL zu erweitern. Es ist jedoch wichtig zu beachten, dass die Entscheidung darüber, welche der potenziellen neuen Hypothesen aus Sicht der Lehre sinnvoll sind, an anderer Stelle getroffen werden muss.

4 Integration des Influence Solvers

In diesem Kapitel wird die Integration des im Rahmen der Bachelorarbeit „An Efficient Algorithm for Proof Search in the Calculus of Influence“ [5] entwickelten Influence Solvers in das iLL beschrieben. Es wird erläutert, wie die Benutzeroberfläche zur Erstellung und Bearbeitung von Experimenten erweitert wird, um die notwendigen Daten für den Influence Solver erstellen zu können. Des Weiteren wird ein Überblick über die Implementierungsdetails des Influence-Solver-Services und die Einbindung der Hypothesenübersetzung im Thesis-Check-Service gegeben. Abschließend werden potenzielle Erweiterungsmöglichkeiten aufgezeigt, mit denen die Menge der mit dem iLL formulierbaren Hypothesen erweitert werden könnte.

Die Codebasis befindet sich auf der Gitlab Instanz des Fachgebites Theoretische Informatik/Formale Methoden in dem Projekt „Bio Tool“ und dem Branch *jan/bachelorThesis*.[1]

4.1 Anlegen eines Experimentes

Wie bereits in Kapitel 2.2 erläutert wurde, bietet das iLL eine Dozierenden-Ansicht, in der Experimente erstellt und bearbeitet werden können. Um den Influence Solver zu integrieren, muss diese Oberfläche überarbeitet werden, damit beim Anlegen oder Bearbeiten eines Experiments auch die notwendigen Daten für den Influence Solver erstellt werden können.

Die Oberfläche, in der ein Experiment erstellt werden kann, besteht aus einem sogenannten horizontalen Stepper, der eine schrittweise Eingabe der Daten ermöglicht. An diesem Grundkonzept wird nichts geändert, allerdings wurden neue Schritte hinzugefügt, einige

[1] https://syre.fm.cs.uni-kassel.de/Georg/bio-tool/-/tree/jan/bachelorThesis

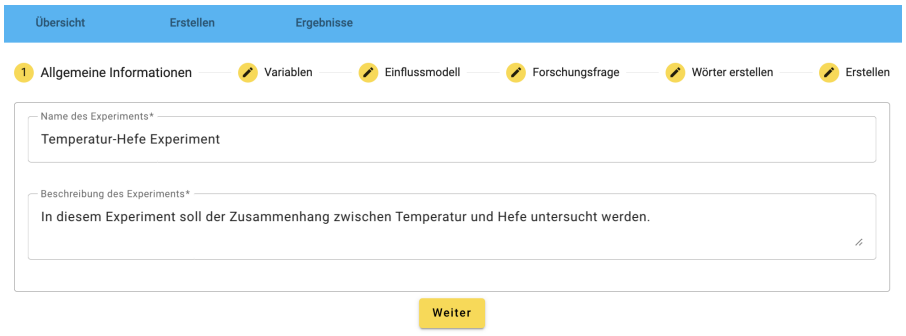


Abbildung 4.1: Schritt 1: Erstellung der allgemeinen Informationen eines Experiments - Benutzeroberfläche zum Anlegen eines Experiments.

alte Schritte wurden überarbeitet, und die Reihenfolge der Schritte wurde von Grund auf neu durchdacht. Die neue Anordnung der einzelnen Schritte ergibt sich aus den formalen Definitionen eines Experiments, insbesondere denen des Calculus of Influence.

Der erste Schritt dient dazu, allgemeine Informationen über das Experiment wie den Namen und eine Beschreibung zu erstellen. Diese Informationen dienen der Identifikation und werden an verschiedenen Stellen in der Anwendung verwendet, beispielsweise in den Übersichtsansichten, in denen alle Experimente aufgelistet werden. Für den Titel wurde ein einfaches Eingabefeld verwendet und für die Beschreibung ein Textfeld, die sich vergrößern lässt, sodass auch längere Beschreibungen gut erstellt werden können. Über den „Weiter“-Button kann zum nächsten Schritt navigiert werden, sofern ein Titel und eine Beschreibung eingegeben wurden. Abbildung 4.1 zeigt den ersten Schritt der Benutzeroberfläche. Hier wurden keine nennenswerten Änderungen vorgenommen.

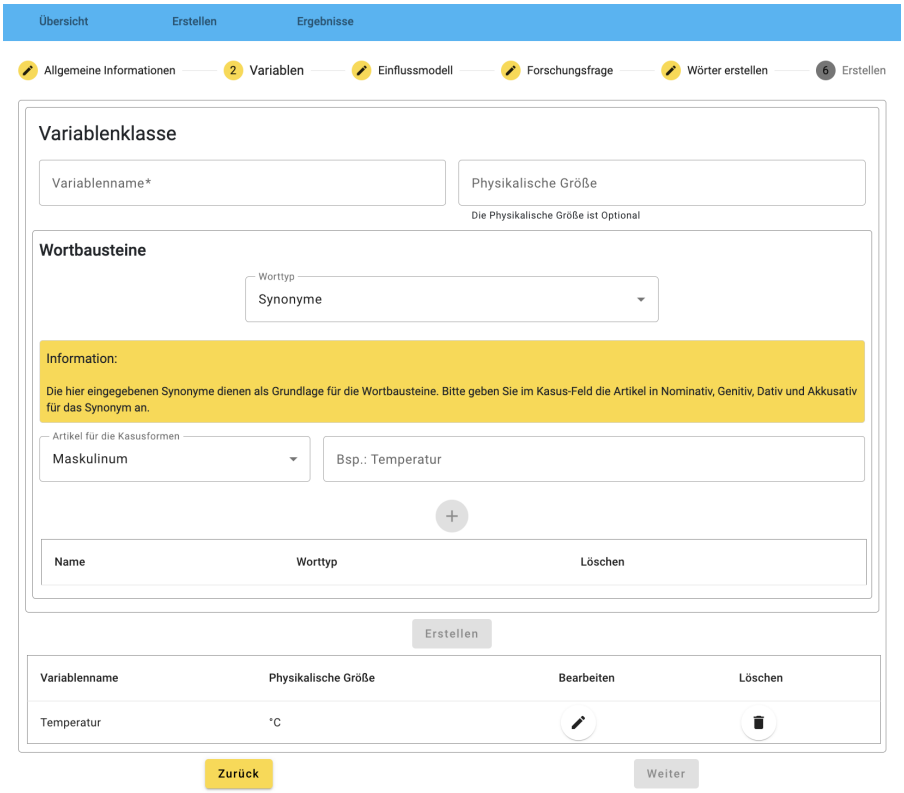


Abbildung 4.2: Schritt 2: Erstellung der Variablen eines Experiments - Benutzeroberfläche zum Anlegen eines Experiments.

Im zweiten Schritt werden die Variablen erstellt. Die Oberfläche ist in zwei Bereiche unterteilt.
Der obere Bereich dient dem Erstellen einer Variablenklasse. Der Variablenname der Variablenklasse wird als Variable für das Influence Model genutzt. Für eine Variablenklasse werden auch Wortbausteine angelegt, die im weitesten Sinne eine praktische Umsetzung der Token aus Kapitel 2.3 darstellen. Es gibt drei verschiedene Wortarten für Variablen, die jeweils einem Worttyp in der Grammatik entsprechen.
Die Synonyme entsprechen den unspezifischen Variablen und werden erstellt, indem eine Artikelform ausgewählt wird und ein Bezeichner eingegeben wird. Der spezifische Wert entspricht der spezifischen Variable und wird als Zahlenwert eingegeben. Für den

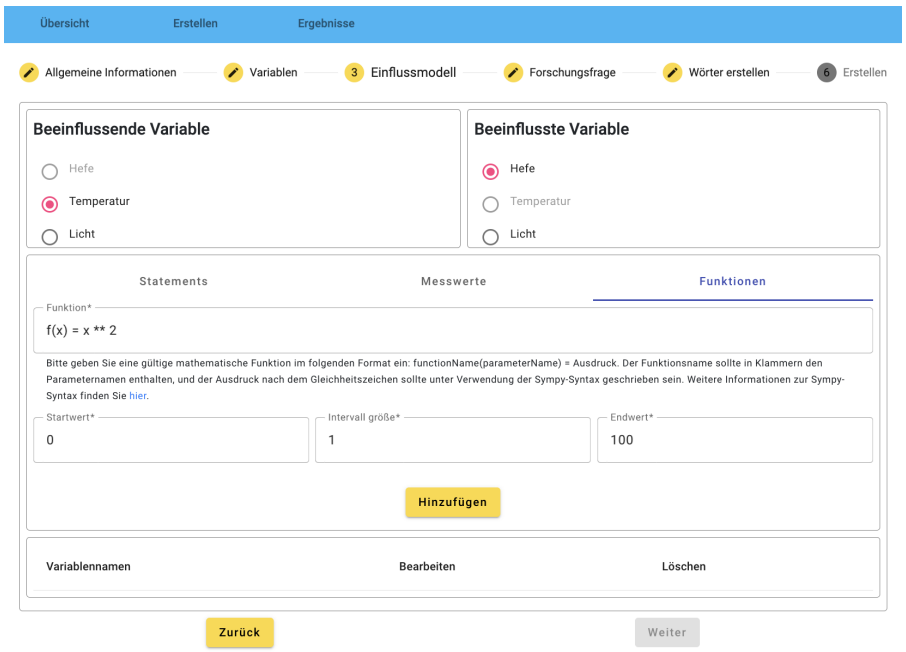


Abbildung 4.3: Schritt 3: Erstellung des Influence Models - Benutzeroberfläche zum Anlegen eines Experiments.

Wertebereich werden eine untere und eine obere Grenze in Form eines Zahlenwertes eingegeben. Beim Erstellen werden drei Wortbausteine erstellt, die die Form [von ... bis ...], [ab ...] und [bis ...] haben. Sowohl für den spezifischen Wert als auch für den Wertebereich wird die physische Größe an die Werte angehängt, sofern eine eingegeben wurde.
Wortbausteine werden durch einen Klick auf den +-Button erstellt, der aktiv wird, sobald eine korrekte Eingabe der Eingabeparameter erfolgt ist. Die erstellten Wortbausteine einer Variablen werden unterhalb des +-Buttons aufgelistet, und es besteht die Möglichkeit, sie zu löschen. Alle Wortbausteine, die innerhalb einer Variablenklasse erstellt wurden, enthalten auch den Variablennamen für das Influence Model.
Mit dem „Erstellen“-Button im unteren Teil der Oberfläche kann die Variablenklasse erstellt werden. Alle erstellten Variablen werden unterhalb des Buttons aufgelistet, und es besteht die Möglichkeit, sie zu bearbeiten oder zu löschen. Abbildung 4.2 zeigt diesen Schritt. Dieser Schritt wurde komplett neu entwickelt.

Nachdem die Variablen im zweiten Schritt angelegt wurden, kann nun im dritten Schritt ein Influence Model für diese erstellt werden (siehe Abbildung 4.3). Die Oberfläche dieses Schrittes besteht aus drei Abschnitten.
Der erste Abschnitt enthält zwei Listen mit jeweils allen Variablen und einem sogenannten Radiobutton für jede Variable. In einer Liste von Radiobuttons ist immer nur eine Option auswählbar. Die Variable, die in der linken Liste ausgewählt wird, entspricht der beeinflussenden Variable, während die Variable in der rechten Liste die beeinflusste Variable repräsentiert. Reflexive Modelle werden im Influence Solver nicht unterstützt, daher ist die Auswahl so implementiert, dass eine Variable, die in einer Liste ausgewählt wurde, in der anderen Liste nicht mehr ausgewählt werden kann.
Der mittlere Abschnitt bietet drei Tabs zur Erstellung des Influence Models. Im ersten

Tab können Statements nach dem Muster „Intervall; Verhalten; Intervall“ eingegeben werden. Ein Intervall besteht aus zwei kommaseparierten Zahlenwerten und für das Verhalten muss einer der vier Strings „mono“, „anti“, „constant“ oder „arbitrary“ gewählt werden. Die Statements werden durch Zeilenumbrüche getrennt. Eine Eingabe könnte zum Beispiel „0, 2; mono ;3.5, 7“ sein.
Der zweite Tab ermöglicht die Eingabe einer Liste von Messwerten. Ein Messwert besteht aus zwei kommaseparierten Zahlenwerten („2,7.9“), wobei der erste Wert der beeinflussenden Variable und der zweite Wert der beeinflussten Variable entspricht. Die einzelnen Messwerte werden durch Zeilenumbrüche getrennt. Der Influence Solver bietet eine Schnittstelle, um aus einer Liste von Zahlenpaaren Statements zu erstellen.
Sowohl die Eingabe der Statements als auch die der Messwerte werden durch reguläre Ausdrücke überprüft, um fehlerhafte Daten zu vermeiden.
Der letzte Tab ermöglicht die Eingabe einer Funktion mit einem Start- und Endwert sowie einer Intervallgröße. Das Intervall legt den Unterschied zwischen aufeinanderfolgenden Werten fest. Bei einer Intervallgröße von eins erhöht sich der Wert um eins. Auch hier erfolgt eine Überprüfung auf Syntaxfehler. Der Funktionsstring wird mithilfe der SymPy-Bibliothek überprüft. Zusätzlich wird geprüft, ob die Start- und Endwerte gültige Zahlenwerte sind und ob die Intervallgröße kleiner als die Differenz des End- und Startwertes ist. Eine Eingabe könnte zum Beispiel „f(x)=x**2+7x-42“, „0“ für den Startwert, „0.1“ für das Intervall und „100“ für den Endwert. Diese Eingabe ergibt den Wertebereich $\{0, 0.1, 0.2, ..., 99.8, 99.9, 100\}$ Im Backend wird aus den eingegebenen Daten eine Liste von Tupeln $(x, f(x))$ erstellt, wobei $f(x)$ die Funktionswerte entsprechend der eingegebenen Funktion darstellt. Diese Liste wird verwendet, um wie bei den Messwerten Statements zu generieren.
Bei der Erstellung eines Modells können alle drei Optionen kombiniert werden. Es erfolgt jedoch keine Überprüfung zwischen den Daten der Tabs, sodass die Nutzer darauf achten müssen, keine widersprüchlichen Daten einzugeben.
Sobald in mindestens einem der Tabs syntaktisch korrekte Daten eingegeben wurden und im oberen Abschnitt zwei Variablen ausgewählt wurden, können die Daten für das Modell über den „Hinzufügen“-Button zu einer Liste im unteren Abschnitt hinzugefügt werden. Ein hinzugefügtes Modell kann entweder bearbeitet oder gelöscht werden, indem die entsprechenden Buttons verwendet werden. Es können mehrere Modelle erstellt werden, wobei zu beachten ist, dass beim erneuten Auswählen von Variablen, für die bereits ein Modell erstellt wurde, das alte Modell überschrieben wird. Auch dieser Schritt wurde komplett neu entwickelt.

Nachdem Modelle für ein oder mehrere Variablenpaare erstellt wurden, kann nun in Schritt vier (siehe Abbildung 4.4) die Forschungsfrage definiert werden (siehe Kapitel 2.1). Die Benutzeroberfläche stellt hierfür wiederum zwei Listen mit Radiobuttons zur Verfügung. In der linken Liste werden alle Variablen aufgeführt, die in einem der zuvor angelegten Modelle eine beeinflussende Variable darstellen. Diese Variablen sind Kandidaten für die unabhängige Variable. In der rechten Liste befinden sich alle Variablen, die in den Modellen beeinflusste Variablen repräsentieren. Sie sind Kandidaten für die abhängige Variable. Wenn es in beiden Listen mehrere Kandidaten gibt, können bei der Auswahl eines Kandidaten in der anderen Liste nur noch die relevanten Kandidaten ausgewählt werden. Dabei werden durch die Berechnung der transitiven Hülle auch Kandidaten von möglichen transitiven Zusammenhängen berücksichtigt.

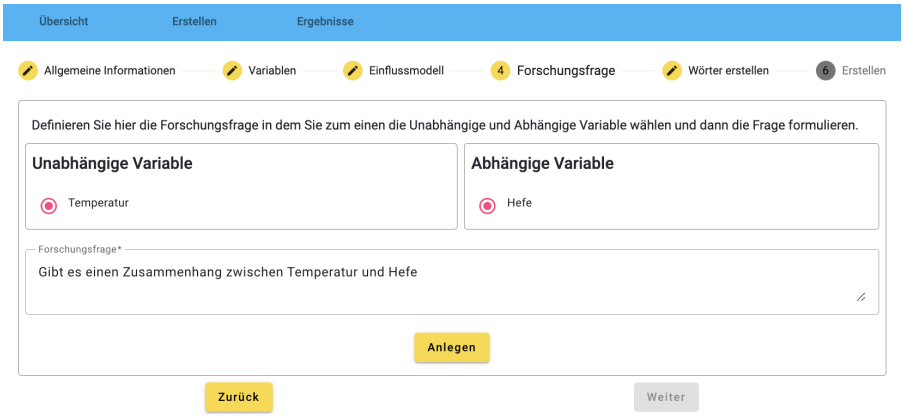


Abbildung 4.4: Schritt 4: Definition der Forschungsfrage - Benutzeroberfläche zum Anlegen eines Experiments.

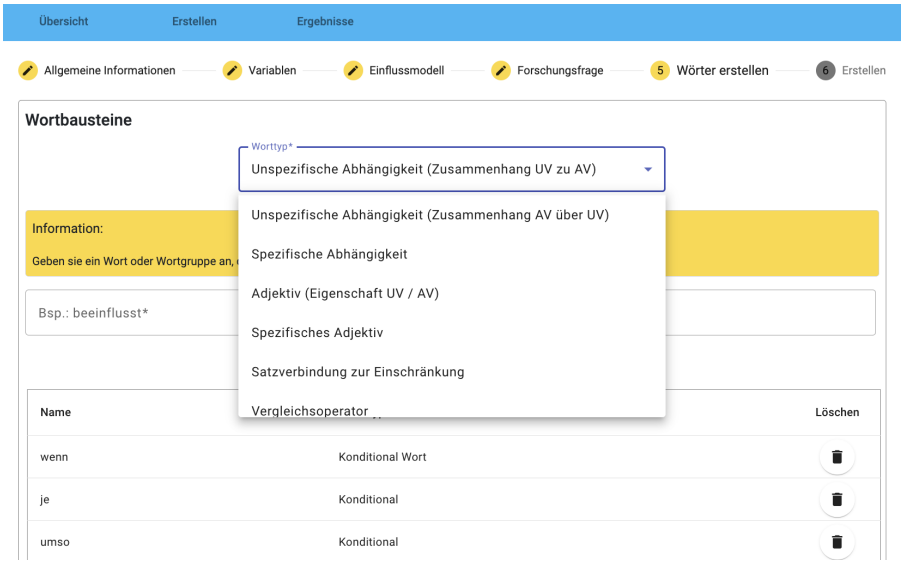


Abbildung 4.5: Schritt 5: Anlegen der Wortbausteine - Benutzeroberfläche zum Anlegen eines Experiments.

Unterhalb dieser beiden Listen befindet sich ein Textfeld, in dem eine textuelle Formulierung der Forschungsfrage Platz findet. Sobald sowohl Variablen ausgewählt wurden als auch eine Frage formuliert wurde, kann diese über einen entsprechenden Button angelegt werden. Eine bereits angelegte Forschungsfrage wird im unteren Bereich der Oberfläche angezeigt. Es besteht die Möglichkeit, sie über einen Button erneut zu bearbeiten. Diesen Schritt gab es bereits zuvor. Er enthielt jedoch nur ein Textfeld, um eine Forschungsfrage zu formulieren.

Im vorletzten Schritt werden die Wortbausteine für die verbleibenden Worttypen erstellt. Die Auswahl eines Worttyps erfolgt über das Select-Menü im oberen Bereich der Oberfläche. Für bestimmte Typen erscheint nach der Auswahl eine Informationsbox mit einer kurzen Erklärung, welche Aspekte beim Anlegen eines Wortbausteins dieses Typs zu beachten sind. Darunter befinden sich entsprechende Input-Felder, die je nach Typ variieren können. Sobald ein Wort über den entsprechenden Button erstellt wird, wird es der Liste am Ende der Oberfläche hinzugefügt. Erstellte Wörter können aus der Liste

auch wieder entfernt werden. Abbildung 4.5 veranschaulicht die Oberfläche. Hier wurde nur das Design überarbeitet und die Worttypen der Variablen entfernt.

Im abschließenden Schritt, der ebenfalls neu ist, befindet sich ein kurzer Infotext sowie ein Button, der das komplette Experiment erstellt und in der Datenbank speichert. Durch Klicken auf den Button wird zur Landingpage weitergeleitet, auf der eine Liste aller Experimente angezeigt wird. Dort besteht die Möglichkeit, ein Experiment zu bearbeiten oder zu löschen.

Die Reihenfolge, in der die Daten für ein Experiment eingegeben werden, baut aufeinander auf. Zuerst werden die Variablen festgelegt, um anschließend für diese Influence Models zu erstellen. Die Forschungsfrage wird dann auf Basis der erstellten Modelle definiert. Abschließend wird das Experiment durch weitere Wortbausteine mit Leben gefüllt, sodass die Hypothesen formuliert werden können.

4.2 Integration im Backend

In diesem Kapitel wird die Implementierung des Influence-Solver-Services und die Integration der Übersetzung im Thesis-Check-Service erläutert. Das Kapitel gibt außerdem einen Überblick über den Aufbau und die Funktionsweise beider Services und zeigt auf, wie sie zusammenarbeiten, um die Hypothesen zu überprüfen und entsprechende Rückmeldungen an das Frontend zu liefern.

Das iLL ist eine Webanwendung, die aus zwei Angular-Frontends[2] besteht, der Dozierenden-Ansicht und der Lernenden-Ansicht. Die Anwendung umfasst auch zwei Backends, die mit dem NestJS-Framework[3] implementiert wurden, den Experiment-Manager-Service und den Thesis-Check-Service. Der Experiment-Manager-Service dient dem Speichern und Bereitstellen von Experimenten und fungiert als Schnittstelle zur Datenbank. Der Thesis-Check-Service empfängt Hypothesen, überprüft sie und speichert sie zusammen mit den Ergebnissen in einer Datenbank.
Der Influence Solver müsste aufgrund seines Funktionsumfangs eigentlich Teil des Thesis-Check-Backends sein. Allerdings wurde dieses Backend in Typescript[4] geschrieben, während der Influence Solver in Python verfasst wurde. Daher wurde im Rahmen dieser Arbeit ein Influence-Solver-Service erstellt, der den Influence Solver um eine Web-API erweitert und eine eigene Datenbank hat. Dieser Service so implementiert, dass er Influence Models speichern und bereitstellen kann sowie Hypothesen in Form eines Statements empfangen und überprüfen kann. Dazu wurden entsprechende Endpunkte implementiert.
Der Programmfluss ist nun wie folgt: Wenn in der Dozierenden-Ansicht ein Experiment erstellt wird, werden die Modelle an den Influence-Solver-Service gesendet, dort aufbereitet und gespeichert. Die restlichen Daten des Experiments werden zum Experiment-Manager-Service gesendet und dort gespeichert. Die Lernenden-Ansicht erhält die Experimente vom Experiment-Manager-Service. Wenn in der Lernenden-Ansicht eine Hypothese erstellt und überprüft wird, wird sie an den Thesis-Check-Service gesendet. Dieser überprüft die Hypothese zunächst auf ihre Syntax, übersetzt sie dann in ein Statement und sendet dieses

[2] https://angular.io/
[3] https://nestjs.com/
[4] https://www.typescriptlang.org/

an den Influence-Solver-Service. Dort wird das Statement überprüft und das Ergebnis an den Thesis-Check-Service zurückgesendet. Schließlich sendet der Thesis-Check-Service das vollständige Ergebnis an das Frontend zurück, wo das entsprechende Feedback angezeigt wird.

Influence-Solver-Service. Der Influence-Solver-Service erweitert den Influence Solver um eine Web-API sowie eine Schnittstelle zu einer MongoDB. Für die Implementierung der Web-API wurde das *FastAPI*-Framework[5] verwendet, während für die Kommunikation mit der Datenbank der Object-Document Mapper (ODM) des Frameworks *mongoengine*[6] eingesetzt wird.

Der in dieser Arbeit entwickelte Controller, der die Web-API implementiert, befindet sich unter `Backend/influence- solver-service/web/controller.py`. Im Folgenden werden die verschiedenen Endpunkte vorgestellt.

Zunächst gibt es den Endpunkt `/influenceExperiments/`, der HTTP-POST-Anfragen entgegennimmt. Diese Anfragen müssen ein JSON-Objekt im Format eines *InfluenceExperimentDto* enthalten. Ein solches Objekt enthält die ID eines Experiments sowie dessen Namen und eine Liste von *InfluenceModelDto*s. Ein *InfluenceModelDto* enthält die beiden Variablennamen und die Rohdaten, die beim Erstellen des Modells für das Experiment verwendet wurden, beispielsweise eine Liste von Statements, Messdaten oder die Daten für eine Funktion. Beide Datenklasse sind unter `Backend/influence-solver-service/web/influenceExperimentDto.py` zu finden.
Der Endpunkt speichert das Data Transfer Object (DTO) in der Datenbank und bereitet die Daten so auf, dass sie vom Influence Solver verwendet werden können. Diese aufbereiteten Daten werden in Form der Datenklasse *InfluenceExperiment* ebenfalls in der Datenbank gespeichert. Die Funktion, die das DTO in ein *InfluenceExperiment* umwandelt, befindet sich unter `Backend/influence-solver-service/web/dtoToModelConverter.py` und die Datenklasse ist in `Backend/influence-solver-service/solver/influenceExperimentModel.py` implementiert.

Über eine HTTP-GET-Anfrage an den Endpunkt `/influenceExperiments/` werden alle *InfluenceExperimentDto*s aus der Datenbank gelesen und als Liste zurückgegeben. Um ein einzelnes *InfluenceExperimentDto* zu erhalten, kann über eine weitere HTTP-GET-Anfrage an den Endpunkt `/influenceExperiments/{experimentID}` gesendet werden, wobei die ID des Experiments Teil der URL ist. Dadurch liest der Service nur dieses spezifische Experiment aus der Datenbank und gibt es als einzelnes *InfluenceExperimentDto* zurück.

Um ein Experiment zu löschen, wird eine HTTP-DELETE-Anfrage an den Endpunkt `/influenceExperiments/{experimentID}` gesendet, wobei die ID des zu löschenden Experiments in der URL enthalten ist.

Des Weiteren stellt der Controller zwei Validierungsendpunkte zur Verfügung. Zum einen gibt es den Endpunkt `/validate/functionExpression`, an den eine *FunctionExpression* mittels HTTP-POST gesendet werden kann. Dieser Endpunkt und die Funktionalität

[5] https://fastapi.tiangolo.com/
[6] http://mongoengine.org/

dahinter wurde im Rahmen dieser Arbeit entwickelt, um den Funktionsstring, der im Frontend eingegeben werden kann, zu überprüfen und dem Frontend mittels eines Validierungs-Objekts mitzuteilen, ob die Syntax korrekt ist. Die Syntaxüberprüfung erfolgt im Backend aus Gründen der Laufzeiteffizienz. Die Funktion wird im Backend verwendet, um zuerst Tupel und dann Statements zu generieren. Daher muss der Funktionsstring der Syntax der *sympy*-Bibliothek[7] entsprechen, da diese Bibliothek zum Parsen und Ausführen des Strings verwendet wird. Die *sympy*-Bibliothek bietet eine solche Überprüfung, wodurch sie die sicherste Wahl darstellt. Die Datenklassen und die Funktion, die die Überprüfung vornimmt, befinden sich im Modul `Backend/influence-solver-service/web/validators.py`.

Der zweite Validierungsendpunkt dient dazu, eine Hypothese in Form eines Statements zu überprüfen. Die URL für diesen Endpunkt lautet `/validate/statement`. Er akzeptiert HTTP-POST-Anfragen, die ein JSON-Objekt im Format eines *StatementValidationObject* enthalten. Intern wird das entsprechende Experiment geladen und ein *Solver* mit diesem initialisiert. Anschließend wird die Methode „solve“ des Solvers mit dem Hypothesen-Statement ausgeführt. Die *Solver* Klasse stammt von Sören Möller. Die Antwort wird als JSON-Objekt in Form eines *ValidationResponse* zurückgegeben. Die Definitionen der Klassen *StatementValidationObject* und *ValidationResponse* befinden sich im Modul `Backend/influence-solver-service/solver/validationModel.py`.
Es ist zu beachten, dass die Intervalle im *Statement*-Objekt optional sind und der Service wurde so implementiert, dass ein nicht angegebenes Intervall als I_∞ interpretiert wird. Da die hier erzeugten Modelle jedoch immer einen endlichen Definitions- und Wertebereich haben, wurde die *adjustIntervals*-Funktion implementiert, die ein Intervall mit der kleinstmöglichen und größtmöglichen Intervallgrenze zu einem gegebenen Experiment erstellt. Diese Intervalle werden dann bei Statements verwendet, bei denen ein oder beide Intervalle fehlen.

Wie bereits zu Beginn dieses Abschnitts erwähnt, wurde für die Kommunikation mit der Datenbank der ODM von *mongoengine* genutzt. Im Controller wird die Verbindung mit der *connect*-Funktion zu den beiden Datenbanken für die DTOs und die Model Datenklassen hergestellt. Des Weiteren wurden in den Modulen *influenceExperimentDtoSchema* und *influenceExperimentSchema* die jeweiligen Schemata implementiert, die wiederum den zuvor erwähnten Datenklassen entsprechen. Zur Vereinfachung der Kommunikation mit der Datenbank werden außerdem die beiden Repository-Module *influenceExperimentDtoRepository* und *influenceExperimentRepository* implementiert, die Funktionen zum Speichern, Lesen und Löschen zur Verfügung stellen. Alle vier Module befinden sich im Verzeichnis `Backend/influence-solver-service/database`.

Übersetzung im Thesis-Check-Service. Im Folgenden wird vereinfacht erklärt, wie eine Überprüfung einer Hypothese abläuft, wobei Implementierungsdetails, die für die Übersetzung nicht von direkter Bedeutung sind, ignoriert werden. Anschließend wird erläutert, wie die Übersetzung in diesen Prozess und die bereits vorhandene Codebasis integriert wurde.

Der Thesis-Check-Service ist, wie bereits erwähnt, mit dem NestJS-Framework implementiert. Über den *ThesisCheckController* stellt er den Endpunkt `/api/checkThesis` zur Ver-

[7] https://www.sympy.org/

fügung, an dem über eine HTTP-POST-Anfrage ein JSON-Objekt im Format einer *Thesis* entgegengenommen wird. Das *Thesis*-Objekt enthält unter anderem die Hypothese in Form einer Liste von *Word*-Objekten. Diese *Word*-Objekte enthalten Informationen wie den Wordtyp sowie den Bezeichner und entsprechen im weitesten Sinne den Tokens aus Kapitel 2.3. Die beiden Datenklassen sind unter den Pfaden `Backend/thesis-check-service/src/thesis-check/dto/thesis.ts` und `Backend/thesis-check-service/src/cross-cutting/data-classes/word.model.ts` zu finden.
Die Hypothese wird anschließend der *validate*-Methode des *ThesisValidator*s übergeben, der eine *ThesisCheckResponse* zurückgibt, die das Ergebnis der Überprüfung enthält. Das Ergebnis wird abschließend in der Datenbank gespeichert, bevor es vom Endpunkt als Antwort zurückgesendet wird.

Im *ThesisValidator* wird die Hypothese auf einen String reduziert, der alle Worttypen der Hypothese enthält. Dieser String wird an einen Parser übergeben, der daraus einen Syntaxbaum auf Basis der Grammatik erstellt. Der Parser wurde mit dem *antlr4ts*-Framework[8] erstellt, das eine TypeScript-Umsetzung des *ANTLR 4* Parser-Generators darstellt. *ANTLR*[9] stellt zusätzlich sogenannte Listener und Visitor zur Verfügung, mit denen der Syntaxbaum durchlaufen werden kann. Bei der Überprüfung der Hypothese kommen zwei Listener zum Einsatz.[10]
Zum einen gibt es den *ThesisSemanticValidator*, der unter anderem rudimentäre semantische Überprüfungen vornimmt und Informationen für ein ausführlicheres Fehlerfeedback erstellt. Zum anderen wird der *ThesisSyntaxValidator* verwendet, der als *ParserErrorListener* fungiert, wenn beim Parsen ein Fehler auftritt. Die Informationen beider Listener werden in einem Objekt gesammelt und am Ende der Überprüfung von der *validate*-Methode zurückgegeben. Alle genannten Klassen befinden sich im Verzeichnis `Backend/thesis-check-service/src/thesis-check/thesis-validation`.

Da die Übersetzung der Hypothesen in Statements auf besagter Grammatik basiert, wurde sie im Rahmen dieser Arbeit in den *ThesisValidator* integriert. Dazu wird ein Visitor genutzt, der für jede Grammatikregel eine Methode implementiert sowie eine allgemeine *visit*-Methode, die den Syntaxbaum als Parameter entgegennimmt und die Schnittstelle nach außen darstellt. Der Visitor durchläuft den Syntaxbaum, indem er die entsprechenden Methoden für die verschiedenen Knotentypen aufruft.
Beispielsweise ruft die Methode *visitHypothese* die Methoden für einen einfachen Zusammenhang (*visitEinfacherZusammenhang*) oder einen komplexen Zusammenhang (*visitKomplexerZusammenhang*) auf, je nachdem, welcher Kontext im Syntaxbaum vorliegt. In den spezifischen Methoden des Visitors wird die Logik für die Übersetzung der Hypothese in Statements implementiert. Der Visitor identifiziert die verschiedenen Bestandteile einer Hypothese wie unspezifische und spezifische Variablen, Intervalle und Abhängigkeiten und erstellt schrittweise die Daten für das *Statement*-Objekt, während der Visitor den Syntaxbaum durchläuft. Der Visitor ist also die praktische Umsetzung der conv-Funktion aus Kapitel 3
Am Ende eines Durchlaufs gibt der Visitor entweder ein *Statement*-Objekt zurück oder *undefined*, wenn sich die Hypothese nicht übersetzen lässt. Der Visitor trägt den Namen *ThesisToStatementConverter* und befindet sich ebenfalls im Verzeichnis

[8] https://github.com/tunnelvisionlabs/antlr4ts
[9] https://www.antlr.org/
[10] [6] bietet einen tieferen Einblick in ANTLR, Listener und Visitors

`.../thesis-validation`.
Eine Sache, die es zu beachten gibt, ist, dass die ANTLR-Tokens aus dem Kontext des Syntaxbaumes nicht direkt genutzt werden können, da sie nur Rückschlüsse auf die Worttypen darstellen. Es wurde eine Methode implementiert, die aus dem Worttyp und der Position im String das entsprechende *Word*-Objekt der eigentlichen Hypothese zuordnet. Das bedeutet für die Hypothese „Die Temperatur beeinflusst den Hefeteig“ gibt es ein Array [(Die Temperatur, UnspezifischeVariable, ...), (beeinflusst, UnspezifischeAbhängigkeit, ...), (den Hefeteig, UnspezifischeVariable, ...)]. Der Parser arbeitet dann auf dem String „UnspezifischeVariableUnspezifischeAbhängkeitUnspezifischeVariable“ und hat keine Informationen über die *Word*-Objekte aus dem Array. Die Methode *stringToArrayPosition* stellt in diesem Zusammenhang wieder eine Verknüpfung her.
Des Weiteren ist anzumerken, dass die Grammatik im Service teilweise andere Namen nutzt und wenige zusätzliche Regeln enthält, die aus theoretischer Betrachtung irrelevant sind und daher abstrahiert wurden. So wird zum Beispiel für die unspezifischen Variablen nochmal zwischen unabhängigen, abhängigen und falschen Unterschieden. Dies ist durch die Integration des Influence Solver eigentlich nicht mehr nötig, wurde aber aufgrund des Fehlerfeedbacks beibehalten.

Die *visit*-Methode des *ThesisToStatementConverter* wird im *ThesisValidator* nach der Überprüfung der beiden zuvor vorgestellten Listener aufgerufen, und zwar nur dann, wenn zuvor keine Fehler aufgetreten sind. Eine fehlerhafte Hypothese lässt sich nicht übersetzen. Falls die Übersetzung erfolgreich ist, wird das Statement an den Influence-Solver-Service gesendet, und die Antwort wird dem *ThesisCheckResponse*-Objekt hinzugefügt. Dieses Objekt wird dann zurück an das Frontend gesendet.

4.3 Erweiterungsmöglichkeiten

Im Wesentlichen gibt es zwei offensichtliche Erweiterungsmöglichkeiten, die sich aus Kapitel 3.3 ableiten. Zum einen ist dies die Erweiterung der Grammatik um Hypothesen, die auf Basis der vorgeschlagenen Formen aus der Analyse der Statements entstanden sind. Hierbei ist es zunächst wichtig, sinnvolle Satzformen zu bilden, die auch mit der deutschen Sprache in Einklang stehen und zu entscheiden, wie sie in die bisherige Struktur der Grammatik integriert werden können.
Für die Erweiterung, die sich durch die Formen des offenen Intervalls ergibt, ist die spezifische Abhängigkeit, um das Verhalten konstant zu erweitern. Die Erweiterungen, die sich durch die Formen des Punktintervalls ergeben, passen ebenfalls gut unter den einfachen Zusammenhang, und zwar als weitere Satzformen des spezifischen Satzes. Hier könnten Hypothesen entworfen werden, die auch für die abhängige Variable spezifische Variablen zulassen.
Die Formen, die für die halb offenen und geschlossenen Intervalle vorgeschlagen wurden, lassen sich als Erweiterung der Einschränkung oder eine zusätzliche Einschränkung für die abhängige Variable integrieren und erweitern so den komplexen Zusammenhang.

Zum anderen ergeben sich Erweiterungen durch das normalisierte Modell. Hierbei müsste der ursprüngliche Influence Solver dahingehend erweitert werden, dass er die Möglichkeit bietet, aus einem herkömmlichen Modell ein normalisiertes Modell zu erzeugen. In der

Datenklasse und dem Schema des *InfluenceExperiment* ist bereits ein Feld dafür angelegt worden, sodass das normalisierte Modell leicht in das Modell des Services integriert werden kann.
Liegt das normalisierte Modell vor, kann die Grammatik um Satzformen für Vergleichs- und lokale Extrema-Hypothesen erweitert werden, wobei diese sicherlich unter den komplexen Zusammenhang fallen. Wird die Grammatik geändert, müssen natürlich im Thesis-Check-Service Anpassungen am *ThesisSemanticValidator* vorgenommen werden, und der *ThesisToStatementConverter* kann um die neuen Übersetzungen und Statements erweitert werden. Auf Seiten des Influence-Solver-Services muss dann entsprechend der `/validate/statement`-Endpunkt für die neuen Statements angepasst werden, und es kann ein Algorithmus zur Überprüfung der neuen Hypothesen für das normalisierte Modell geschrieben werden.

Es gilt zu beachten, dass eine Erweiterung der Grammatik um neue Worttypen/Terminale auch Anpassungen im Frontend mit sich bringt.

5 Fazit und Ausblick

Das vorliegende Kapitel zieht ein Fazit über die Ergebnisse der Arbeit und gibt einen Ausblick auf mögliche Weiterentwicklungen des iLLs.

Die erste Forschungsfrage, die nach der Überführung von Hypothesen in Statements fragt, wird in Kapitel 3 beantwortet. Die entwickelte conv-Funktion ermöglicht diese Übersetzung. Die Funktion ist, basierend auf den Ableitungsregeln der Grammatik, in der Lage, die meisten Hypothesen zu übersetzen. Dabei wird auch aufgezeigt, dass es in einigen Fällen Einschränkungen bei der Übersetzung gibt und woran dies liegt. Die Umsetzung der Übersetzung wurde erfolgreich im iLL implementiert. Der Influence Solver wurde über eine Web-API in das iLL integriert und in den Prozess der Hypothesenüberprüfung eingebunden. Des Weiteren wurden Möglichkeiten aufgezeigt, wie sich durch die Integration auch die Grammatik erweitern lässt.

Die zweite Forschungsfrage, die nach einer strukturierten Methode zum Anlegen von Experimenten fragt, wird in Kapitel 4 beantwortet und wurde in der Dozierenden-Ansicht implementiert. Die gewählte Struktur zum Anlegen eines Experiments in der Dozierenden-Ansicht wurde aus den formalen Definitionen eines Experiments abgeleitet. Dadurch wurde das Erstellen eines Experiments zu einem Prozess von aufeinander aufbauenden Schritten: Die Daten werden nacheinander erfasst, bis am Ende alle Daten vorliegen, die für ein Experiment benötigt werden.

Das iLL verfügt nun über die Fähigkeit, nicht nur die syntaktische, sondern auch eine semantische Überprüfung der Hypothesen durchzuführen. Das Ergebnis dieser Überprüfung fließt in das Feedback für die Lernenden mit ein.

In den Kapiteln 3 und 4 wurden bereits einige Möglichkeiten für Erweiterungen genannt, die darauf basieren, die Grammatik für neue Formen von Hypothesen zu erweitern, die sich durch die Integration des Influence Solvers ergeben.
Zwei Möglichkeiten, die das Erstellen eines Experiments in der Dozierenden-Ansicht verbessern, sind zum einen eine grafische Darstellung der erstellten Influence Models. Dadurch könnte ein erstelltes Modell leicht von den Nutzern auf Fehler überprüft werden. Es existiert bereits eine Funktion im Programmcode von Sören Möller, die Modelle plotten kann, was als Grundlage für die Umsetzung dieser Erweiterung dienen könnte.
Zum anderen kann im Rahmen der Modellerstellung die Möglichkeit integriert werden, spezielle Punkte zu definieren, die für das Experiment von besonderer Bedeutung sind, etwa lokale Extrema oder ähnliches. Die Einbindung in das Modell könnte durch eine individuelle Statement-Modellierung umgesetzt werden.

Abschließend lässt sich sagen, dass diese Arbeit das iLL um ein bedeutendes Feature erweitert hat und den Funktionsumfang erweitert hat. Zusätzlich wurden weitere Möglichkeiten zur Verbesserung des Systems aufgezeigt.

A Anhang

hypothese →
  einfacherZusammenhang | *komplexerZusammenhang*
einfacherZusammenhang →
  unspezifischerSatz | *spezifischerSatz* | *konditionalSatz*
komplexerZusammenhang →
  vergleichSatz | *einfacherZusammenhang einschränkungsSatz*
unspezifischerSatz →
  UnspezifischeVariable UnspezifischeAbhängigkeit UnspezfischeVariable
spezifischerSatz →
  abhängigeVariableSpezifischeAbhängigkeit | *spezifischeAbhängigkeitAbhängigeVariable*
abhängigeVariableSpezifischeAbhängigkeit →
  UnspezifischeVariable SpezifischeAbhängigkeit Temporal? Bei SpezifischeVariable |
  UnspezifischeVariable SpezifischeAbhängigkeit Temporal? Bei Quantität?
  UnspezifischeVariable Temporal?
spezifischeAbhängigkeitAbhängigeVariable →
  Bei SpezifischeVariable SpezifischeAbhängigkeit UnspezifischeVariable Temporal? |
  Bei Quantität? UnspezifischeVariable SpezifischeAbhängigkeit UnspezifischeVariable
konditionalSatz →
  Konditional UnspezifischeVariable (Quantität | Temporal)? SpezifischeAbhängigkeit
  Konditional SpezifischeAbhängigkeit UnspezifischeVariable (Quantität|Temporal)? |
  Konditional (Quantität|Temporal)? UnspezifischeVariable SpezifischeAbhängigkeit
  Konditional (Quantität|Temporal)? SpezifischeAbhängigkeit UnspezifischeVariable
vergleichSatz →
  Quantität UnspezifischeVariable UnspezifischeAbhängigkeit UnspezifischeVariable
  VergleichsOperator Quantität UnspezifischeVariable |
  UnspezifischeVariable SpezifischeAbhängigkeit Bei SpezifischeVariable
  VergleichsOperator Bei SpezifischeVariable
einschränkung →
  Konnektor Wertebereich

Abbildung A.1: Grammatik zum Ableiten von Hypothesen

Abbildungsverzeichnis

Literatur

[1] Florian Bruse, Martin Lange und Sören Möller. „Formal Reasoning about Influence in Natural Sciences Experiments“. Paper, Universität Kassel, Theoretische Informatik/Formale Methoden.

[2] Florian Bruse u. a. „The Calculus of Temporal Influence“. zur Veröffentlichung akzeptiert, Universität Kassel, Theoretische Informatik/Formale Methoden.

[3] Marit Kastaun u. a. „ProfiLL–Professionalisierung durch intelligente Lehr-Lernsysteme“. In: *Bildung, Schule und Digitalisierung* (2020), S. 357–363.

[4] Lukas Mentel. „Categories of Biological Experiments“. Seminararbeit, Universität Kassel.

[5] Sören Möller. „An Efficient Algorithm for Proof Search in the Calculus of Influence“. Bachelorarbeit, Universität Kassel, Theoretische Informatik/Formale Methoden.

[6] Terence Parr. „The definitive ANTLR 4 reference“. In: *The Definitive ANTLR 4 Reference* (2013), S. 1–326.

[7] Uwe Schöning. *Theoretische Informatik-kurz gefasst*. Springer, 1992.