Interaktives Debugging von Wissensbasen[1]

Patrick Rodler[2]

Abstract: In heutigen Zeiten, wo automatisierte intelligente Applikationen aus unserem Leben kaum noch wegzudenken sind, ist es von entscheidender Bedeutung, dass die solchen Systemen zugrunde liegenden Wissensbasen hohen Qualitätsanforderungen gerecht werden. Als Kurzbeschreibung der gleichnamigen Dissertation motiviert dieser Beitrag den Einsatz von interaktivem Debugging von Wissensbasen durch das Aufzeigen der Probleme existierender nicht-interaktiver Systeme. Es wird ein generisches interaktives Debuggingsystem beschrieben und dessen Funktionsweise erläutert. Zudem werden einige konkrete Anwendungen solcher Debuggingverfahren anhand praktischer Use Cases aus der realen Welt skizziert. Schließlich werden die konkreten Kontributionen der Dissertation vermittelt, u.a. die erstmalige Entwicklung bewiesen korrekter, vollständiger, optimaler und für jegliche monotone Logik anwendbarer Verfahren zur Lösung verschiedener praxisrelevanter Debuggingprobleme.

1 Motivation

Anwendungen der künstlichen Intelligenz (KI) sind im letzten Jahrzehnt zu einem allgegenwärtigen und ständigen Begleiter des Menschen geworden und können sowohl im Alltag (z.B. Smart Home, Empfehlungssysteme sowie Konfigurationssysteme im eCommerce) als auch in Unternehmen (z.B. Planungssysteme für Transportlogistik, Industrie 4.0) und kritischen Systemen (z.B. medizinische Expertensysteme zur Diagnose- bzw. Therapiefindung, Roboter in der Raumfahrt, eHealth Anwendungen) maßgeblich zur Komfortabilität, zur Sicherheit sowie zur Effizienz von Prozessen und Tätigkeiten beitragen. Zentral für viele KI Anwendungen ist die Repräsentation und die Verarbeitung von Wissen. Einen Hauptansatz hierfür stellen Ontologien bzw. Wissensbasen (kurz WBn) formuliert durch logikbasierte Sprachen dar. Entsprechend kodiert können WBn von Computersystemen automatisiert verarbeitet und genutzt werden. Die Formulierung der WBn ist jedoch in den meisten Fällen die Aufgabe eines menschlichen Operators. Beispiele logischer Sprachen, die für solche Zwecke herangezogen werden, sind Aussagenlogik, Datalog, Prädikatenlogik erster Ordnung, die Web Ontology Language (OWL) oder verschiedene Beschreibungslogiken. Aufgrund des erfolgreichen Einsatzes wissensbasierter Systeme findet man heutzutage in unterschiedlichsten Branchen wie Biologie, Geologie, Medizin, Chemie oder der Industrie Experten, die versuchen mittels logischer WBn zur Lösung verschiedenster Problemstellungen zu gelangen. Die Größe, der Informationsgehalt und die Komplexität der so konzipierten WBn sind dabei stetig wachsend. Nunmehr sind WBn mit mehreren Hunderttausenden logischen Sätzen keine Seltenheit mehr.[3] Solche WBn bedeuten jedenfalls eine signifikante Herausforderung für die in deren Entstehung, Wartung, Anwendung und Qualitätssicherung involvierten Personen und Tools.

[1] Englischer Titel der Dissertation: “Interactive Debugging of Knowledge Bases”, siehe [Ro15]

[2] Institut für Angewandte Informatik, Alpen-Adria Universität Klagenfurt, patrick.rodler@aau.at

[3] Siehe Abschnitt 6.

Den essentielle Nutzen einer WB stellt die automatische Schlussfolgerung implizit durch die WB gegebenen Wissens sowie die Beantwortung komplexer Fragestellungen betreffend der durch die WB modellierten Domäne dar. Die erfolgreiche Realisierung dieser Services setzt allerdings voraus, dass die entsprechende WB das Mindestqualitätskriterium der logischen Konsistenz erfüllt. Denn aus einer inkonsistenten WB können *beliebige* Schlussfolgerungen deduziert werden, womit auch deren Nutzen für jegliche wissensbasierte Anwendung schwindet. Zusätzlich zur Konsistenz können weitere *Anforderungen* an die WB gestellt werden. Beispielsweise könnte Kohärenz postuliert werden, was bedeutet, dass aus einer (prädikatenlogischen) WB für kein darin enthaltenes Prädikat folgen darf, dass dieses für alle möglichen Instanzierungen falsch sein muss. Zusätzlich können konkrete Test Cases spezifiziert werden, welche Auskunft über gewünschte (*positive Test Cases*) bzw. ungewünschte (*negative Test Cases*) Schlussfolgerungen aus der korrekten WB geben. Dieses Test Case Paradigma kann als Analogon zum Softwaredebugging betrachtet werden, wo Test Cases eingesetzt werden, um die korrekte Semantik des Programmcodes zu verifizieren.

Je größer und komplexer WBn werden, desto höher wird die Gefahr, dass zuvor genannte Anforderungen oder Test Cases verletzt werden und die WB damit nicht mehr den notwendigen Qualitätskriterien gerecht wird. Fehlerhafte WBn entstehen häufig aufgrund dessen, dass die menschliche Kognition ab einem gewissen (relativ geringen) Grad an Komplexität (siehe Beispiel auf Seite 3) bereits nicht mehr in der Lage ist, das spezifizierte Wissen zu überblicken, zu verarbeiten und die daraus entstehenden Implikationen zu erkennen, geschweige denn auf deren Richtigkeit hin zu prüfen. Zudem haben Studien der kognitiven Psychologie (z.B. [CP71]) gezeigt, dass Menschen beim Formulieren und Interpretieren logischer Formalismen systematische Fehler begehen. Weitere Gründe für die Nichtkonformität einer WB mit den geforderten Eigenschaften sind darin zu finden, dass WBn oftmals durch eine Vielzahl von Personen in verteilter und größtenteils autonomer Art und Weise konstruiert werden, was die Entstehung von Widersprüchen in der WB begünstigt. Das *OBO Project*[4] und der *NCI Thesaurus*[5] sind Beispiele solcher kollaborativer Entwicklungsprojekte von WBn. Der Einsatz automatischer Systeme zur Generierung von (Teilen von) WBn kann diese Problematik zusätzlich signifikant verschlimmern [Me11]. Aus all diesen Gründen ist es essentiell, skalierbare Methoden zu entwickeln, die die effiziente Fehlerfindung und -korrektur in WBn ermöglichen.

2 Nicht-interaktives Debugging von Wissensbasen

Gegeben bestimmte Anforderungen und Test Cases kann durch WB Debugging Methoden ein (potentieller) Fehler in einer WB K lokalisiert werden. Dies geschieht mittels Berechnung einer Teilmenge D der logischen Sätze in K. Eine solche (potentiell fehlerhafte) Teilmenge nennen wir eine *Diagnose*. Mindestens alle Sätze, die Element einer ($\subseteq$-)minimalen Diagnose sind, müssen geeignet modifiziert oder aus der WB entfernt werden, um eine *Lösungs-WB* K^* zu erhalten, die alle postulierten Anforderungen und Test Cases erfüllt.

[4] Siehe http://obo.sourceforge.net
[5] Siehe http://nciterms.nci.nih.gov/ncitbrowser

Als Input erwartet ein WB Debugging System (WBDS) eine sogenannte *Diagnoseproblem-instanz (DPI)*, welche durch folgende Parameter gegeben ist: (1) Eine WB K, die über einer beliebigen monotonen logischen Wissensrepräsentationssprache $\mathscr{L}$ formuliert ist. Alle logischen Sätze in K können korrekt oder fehlerhaft sein. (2) Optional: Eine WB B (über $\mathscr{L}$), welche bestimmtes, bereits anerkanntes oder auf Richtigkeit geprüftes Hintergrundwissen über die durch K repräsentierte Domäne spezifiziert. Alle logischen Sätze in B sind per Annahme korrekt. (3) Eine Menge R von Anforderungen an die korrekte WB. (4) Mengen von positiven (P) sowie negativen (N) Test Cases (über $\mathscr{L}$), welche die geforderten semantischen Eigenschaften der korrekten WB angeben. (5) Optional: Meta-informationen **FP**, z.B. in Form von Fehlerwahrscheinlichkeiten einzelner Sätze in K.

Ein WBDS benötigt zudem ein logisches Schlussfolgerungssystem (SFS), welches die Entscheidung über Konsistenz (oder Kohärenz) sowie die Berechnung von logischen Folgerungen einer Menge logischer Sätze über $\mathscr{L}$ ermöglicht. Ist nun ein DPI sowie ein adäquates SFS gegeben, so liegt der Fokus eines WBDS auf (einer Teilmenge aller) möglichen Fehlerkandidaten, gewöhnlich die Menge der ($\subseteq$-)*minimalen Diagnosen*. Im Standardfall wird diejenige reparierte Lösungs-WB K^* zurückgegeben, die mithilfe der wahrscheinlichsten oder kardinalitätsminimalen aller berücksichtigten Diagnosen aus der ursprünglichen WB berechnet wird.

3 Probleme des nicht-interaktiven Debuggings von Wissensbasen

Eingesetzt zur Lösung von Problemen der realen Welt müssen WBDSe häufig mit einer enormen Anzahl (häufig tausende) von Lösungs-WBn umgehen. Dabei haben je zwei durch unterschiedliche minimale Diagnosen erzeugte Lösungs-WBn unterschiedliche Semantik hinsichtlich der implizierten und nicht-implizierten logischen Sätze. Selektiert man jedoch einfach eine beliebige der möglichen Lösungs-WBn und ist diese falsch im Sinne ungewünschter Semantik so kann dies zu unerwarteten Implikationen und Nicht-Implikationen, verlorenen gewünschten Folgerungen und überraschenden Fehlern während der weiteren Entwicklung der WB führen. Eine manuelle Begutachtung einer großen Menge an Diagnosen oder Lösungs-WBn ist sehr zeitaufwändig (sofern in der Praxis überhaupt zumutbar), fehleranfällig und in vielen Fällen aufgrund der Berechenbarkeitskomplexität der Diagnosen schlichtweg nicht realisierbar.

Überdies wurden einige WBDSe von [St08] eingehend getestet – mit einem ziemlich ernüchternden Resultat. Und zwar zeigten die meisten WBDSe schwerwiegende *Performanzprobleme*, verursachten einen *Hauptspeicherüberlauf*, könnten nicht alle in der WB vorhandenen Fehler extrahieren (*Unvollständigkeit*), wiesen korrekte Teile der WB als fehlerhaft aus (*Unrichtigkeit*), retournierten *nur triviale Lösungen* oder wiesen nicht-minimale Diagnosen als minimal aus (*Nicht-Minimalität*).

Ein motivierendes Beispiel. Man betrachte folgende in Prädikatenlogik erster Ordnung gegebene WB aus der Domäne der Anatomie:

$$\forall X(sehne(X) \leftrightarrow \forall Y(verbindetKnochenMit(X,Y) \rightarrow muskel(Y))) \quad (1)$$

$$\forall X((\exists Y\, verbindetKnochenMit(X,Y)) \rightarrow sehne(X)) \quad (2)$$

$$\forall X(band(X) \rightarrow \neg sehne(X)) \tag{3}$$
$$band(kreuzband) \tag{4}$$

Natürlichsprachlich formuliert sagt diese WB in Reihenfolge der angeführten logischen Sätze folgendes aus: (1) Etwas ist eine Sehne genau dann, wenn es einen Knochen ausschließlich mit einem Muskel verbinden kann. (2) Alles, das einen Knochen mit etwas verbindet, ist eine Sehne. (3) Kein Band ist zugleich eine Sehne (und umgekehrt). (4) Das Kreuzband ist ein Band.

Diese WB ist tatsächlich inkonsistent, erfüllt also die Minimalanforderung nicht. Der Leser wird möglicherweise zustimmen, dass diese Inkonsistenz trotz sehr kleiner WB nicht leicht erkennbar ist. Selbst sehr erfahrene Logiker, nämlich Entwickler von Schlussfolgerungssystemen, sind daran gescheitert, dies anhand einer diesem Beispiel strukturell sehr ähnlichen WB einzusehen. Konsultiert man nun für diese WB ein WBDS, so ergibt sich schnell, dass jeder logische Satz (1) - (4) in der WB eine minimale Diagnose ist. Aufgrund fehlender Zusatzinformation kann das WBDS nun lediglich eine dieser Diagnosen (bzw. eine daraus generierte Lösungs-WB) ausgeben, z.B. (3). Durch Entfernung des logischen Satzes (3) wird die WB wieder konsistent und das WB Debugging Problem ist gelöst – vorerst. Formuliert der Benutzer nun einen neuen logischen Satz (3'): $\neg sehne(kreuzband)$, um die durch die Löschung von (3) verloren gegangene gewünschte Implikation, dass das Kreuzband keine Sehne ist, zu kompensieren, erhält er plötzlich wieder eine inkonsistente WB und eine neue Debuggingsession ist nötig. Die Reparatur der WB war also nicht nachhaltig.

4 Die Lösung: Interaktives Debugging von Wissensbasen

Algorithmen zum Interaktiven Debugging von WBn zielen darauf ab, die Menge der Lösungs-WBn durch Benutzerinteraktion sukzessive zu verkleinern. Anders formuliert versuchen diese Verfahren durch regelmäßiges Beschneiden des Lösungssuchraums, repräsentiert durch einen Suchbaum, diesen in seiner Größe handhabbar zu halten und damit Performanzprobleme bzw. Speicherüberläufe zu vermeiden. *Der Benutzer* bezieht sich in diesem Fall entweder auf eine (Gruppe von) Person(en) oder ein geeignetes automatisches Orakel (z.B. ein Wissensextraktionssystem). Man nimmt dabei an, dass der Benutzer zumindest über eine teilweise Expertise in der von der WB modellierten Domäne verfügt. Während einer interaktiven Diagnosesession werden dem Benutzer automatisch generierte und optimierte Anfragen über die Domäne, welche durch die fehlerhafte WB eigentlich beschrieben werden sollte, zur Beantwortung vorgelegt. Die Konstruktion einer solchen Anfrage setzt die Vorausberechnung einer aus mindestens zwei Elementen bestehenden Teilmenge $\mathbf{D}$ aller minimalen Diagnosen voraus. Wir bezeichnen $\mathbf{D}$ als *führende Diagnosen*. Jede Anfrage ist eine Menge oder Konjunktion von logischen Sätzen, die Schlussfolgerungen einer korrekten Teilmenge der WB sind. Eine korrekte Teilmenge der WB ist eine, die weder die gegebenen Anforderungen R noch die gegebenen Test Cases P, N verletzt. Hinsichtlich einer bestimmten Anfrage Q kann jede Menge minimaler Diagnosen der WB, insbesondere also $\mathbf{D}$, in drei Teile partitioniert werden. Der erste Teil ($\mathbf{D}^+(Q)$) inkludiert alle minimalen Diagnosen, die *nur* in Übereinstimmung mit der positiven Beantwortung der Anfrage stehen, der zweite ($\mathbf{D}^-(Q)$) all jene, die *nur* in Übereinstimmung

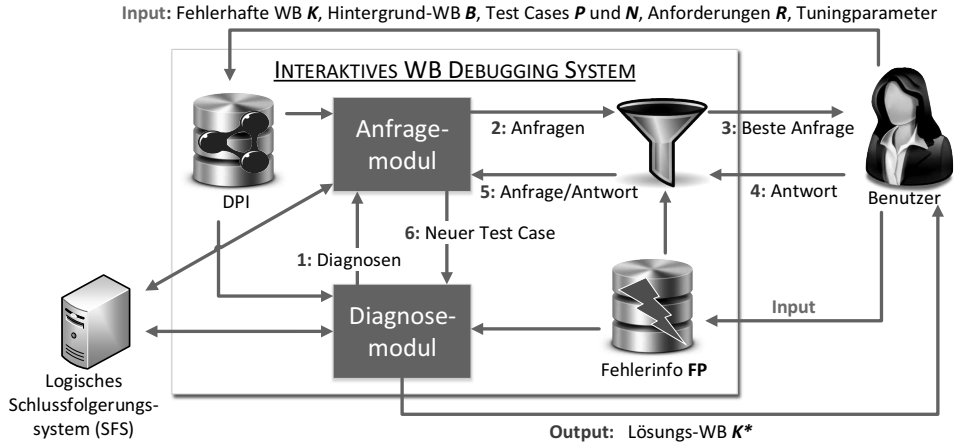

Abb. 1: Das Prinzip des interaktiven Debuggings von Wissensbasen.

mit der negativen Beantwortung stehen, und der dritte ($\mathbf{D}^0(Q)$) all jene, die mit beiden Antworten konsistent sind. Ein positiv beantwortetes Q signalisiert, dass die Konjunktion der logischen Sätze in Q eine Implikation der korrekten WB sein muss. Daher wird ein solches Q zur Menge der positiven Test Cases P hinzugefügt. Analog wird ein vom Benutzer negiertes Q – ein Indiz dafür, dass mindestens ein logischer Satz in Q nicht aus der korrekten WB folgen soll – der Menge der negativen Test Cases N zugeordnet.

Die Allokation einer Anfrage Q zu einer der beiden Test Case Mengen resultiert in einem völlig neuen Debuggingszenario. Für das entstandene neue WB Debugging Problem sind alle Elemente von $\mathbf{D}^-(Q)$ keine minimalen Diagnosen (mehr), sofern Q als positiver Test Case klassifiziert wurde. Andernfalls werden alle Diagnosen in $\mathbf{D}^+(Q)$ invalidiert. Auf diese Art kann der Benutzer durch sequentielle Beantwortung von systemgenerierten Anfragen den Suchraum auf eine einzige noch mögliche minimale Diagnose D_t, also die einzige noch nicht ausgeschlossene minimal invasive Änderung der falschen WB, reduzieren. Die durch die Löschung dieser Diagnose D_t aus der ursprünglichen WB K und durch Hinzufügung des Hintergrundwissens sowie der Konjunktion aller in positiven Test Cases vorkommenden logischen Sätzen reparierte WB $(K \setminus D_t) \cup (\bigcup_{p \in P} p)$ ist diejenige unter allen Lösungs-WBn, die exakt der geforderten Semantik entspricht. Man beachte, dass die Aufnahme der logischen Sätze in positiven Test Cases dabei als Ersatz gewünschter Implikationen dient, die durch die Entfernung der logischen Sätze in der Diagnose aus der WB nicht mehr gegeben wären.

Die Vorteile des interaktiven Debuggings lassen sich also wie folgt zusammenfassen: Der Benutzer kann die fehlerhafte WB debuggen *ohne* analysieren zu müssen, welche Implikationen durch die fehlerhafte WB gegeben oder nicht gegeben sind, warum bestimmte Implikationen durch die fehlerhafte WB gegeben oder nicht gegeben sind oder warum die WB überhaupt fehlerhaft ist, *indem* er/sie Anfragen beantwortet, ob bestimmte (einfache) Aussagen in der gewünschten, zu modellierenden Domäne gelten oder nicht gelten sollen, *unter Garantie* der Auffindung der Lösungs-WB mit exakt der gewünschten Semantik.

5 Design und Funktion eines Interaktiven Debugging Systems

Das Schema eines Interaktiven WB Debugging Systems (IWBDS) wird durch Abbildung 1 veranschaulicht. Wie ein WBDS erhält auch ein IWBDS eine Diagnoseprobleminstanz (DPI) als Eingabe und beruht auf der Verfügbarkeit eines entsprechenden Schlussfolgerungssystems (SFS) für die Logik $\mathscr{L}$, in der die fehlerhafte WB formuliert wurde. Im Falle des IWBDS gibt es noch eine Reihe weiterer Eingabeparameter, mit Hilfe derer man das Verhalten des Systems entlang mehrerer Dimensionen adjustieren kann. Wir nennen diese zusätzlichen Parameter *Tuningparameter*. Diese ermöglichen beispielsweise die Beeinflussung der Reaktionszeit des Systems (d.h. die Zeit zwischen Eingabe einer Anfrageantwort und der Bereitstellung der nächsten Anfrage), die Wahl eines bestimmten Gütemaßes zur Anfrageselektion, die Angabe einer Lösungsfehlertoleranz σ (d.h. das Finden von approximativen Lösungen unterschiedlicher Güte) oder die Auswahl einer Strategie, mittels derer der Suchraum exploriert und beschnitten wird.

Der durch Abbildung 1 illustrierte Ablauf (siehe mit 1 - 6 annotierte Pfeile) während einer interaktiven Debuggingsession ist folgender: (1) Eine Menge **D** von führenden Diagnosen wird durch das Diagnosemodul (unter Zuhilfenahme der Fehlerinformation **FP**, sofern verfügbar) durch Nutzung des SFS berechnet und an das Anfragemodul weitergegeben. (2) Das Anfragemodul erzeugt unter Ausnutzung der führenden Diagnosen **D** einen Pool von möglichen Anfragen und übergibt diesen an die Anfrageselektion. (3) Die Anfrageselektion filtert die „beste“ Anfrage Q (gemäß einem Gütemaß, oftmals unter Berücksichtigung der Fehlerinformation **FP**, sofern verfügbar) heraus und zeigt diese dem Benutzer an. (4) Der Benutzer gibt seine Antwort auf Q ein (kennt er diese nicht, so fordert er – evtl. mehrmals – mittels „überspringen“ eine Alternativanfrage an). (5) Die Anfrage Q gemeinsam mit der eingegebenen Antwort wird zur Formulierung eines entsprechenden neuen Test Cases verwendet. (6) Dieser neue Test Case wird an das Diagnosemodul zurückgeliefert und in den folgenden Iterationen mitberücksichtigt. Wenn das Stoppkriterium (ermittelt durch die Fehlertoleranz σ, siehe oben) nicht erfüllt ist, also im Moment kein bekannter Lösungskandidat genügend wahrscheinlich ist, startet eine weitere Iteration bei Schritt 1. Ansonsten wird die durch die aktuell höchstwahrscheinliche minimale Diagnose konstruierte Lösungs-WB K^* zurückgegeben.

Ein motivierendes Beispiel (Fortsetzung). Erinnern wir uns an das auf Seite 3 diskutierte Beispiel. Wird zur Auflösung der Inkonsistenz der WB nun ein IWBDS herangezogen, so könnte die erste Anfrage beispielsweise lauten: „Soll jede Instanz in der gewünschten Domäne eine Sehne sein?“ Wenn diese Anfrage verneint wird, da der Benutzer weiß, dass es z.B. auch den Bizeps gibt, der ein Muskel ist, so kann das IWBDS daraus deduzieren, dass der Fehler in (1) oder (2) liegen muss, da die verneinte Anfrage $\forall X\, sehne(X)$, also der neue negative Test Case, eine Implikation dieser beiden logischen Sätze ist. Die nunmehr verbleibenden minimalen Diagnosen sind also (1) und (2). Nach einer zweiten Anfrage, etwa: „Kann etwas einen Knochen mit etwas verbinden ohne selbst eine Sehne zu sein?“, die der Benutzer bejaht, z.B. da er als Anatomieexperte weiß, dass auch ein Band einen Knochen mit etwas (nämlich einem anderen Knochen) verbinden kann, wird der Suchraum auf eine einzige exakte Lösung reduziert. Diese verlangt nach einer geeigneten Korrektur bzw. der Löschung von Satz (2) aus der WB. Der Benutzer könnte beispielsweise nach Analyse

von (2) herausfinden, dass die Negation vor dem Satz vergessen wurde, der richtige Satz (2) also $\neg\forall X((\exists Y verbindet Knochen Mit(X,Y)) \rightarrow sehne(X))$ lauten sollte.

6 Konkrete Anwendungsbeispiele des Interaktiven Debuggings

Ein Exempel eines (Meta-)Repositoriums von WBn ist die *Open Ontology Repository*,[6] welche unter anderem zur prominenten Ontologiedatenbank *Bioportal*[7] verlinkt. Letztere beinhaltet zum Beispiel die Ontologie *SNOMED CT (Systematized Nomenclature of Medicine – Clinical Terms)*, die umfassendste, mehrsprachige medizinische Terminologie der Welt. Sie beinhaltet aktuell mehr als 311.000 medizinische Terme und über 1.3 Millionen logische Sätze, die Zusammenhänge zwischen diesen Termen auf formale und eindeutige Weise spezifizieren, und wird in mehr als 50 Ländern der Welt in eHealth Applikationen (z.B. elektronische Gesundheitsakte) eingesetzt.[8] Aufgrund der zum Teil verteilten Entwicklung und der Komplexität und Größe von SNOMED-CT entstehen (laufend) eine Vielzahl von logischen Widersprüchen und diversen anderen ungewünschten semantischen Eigenschaften.

Eine andere, in vielen Anwendungen eingesetzte WB, der *NCI (National Cancer Institute) Thesaurus*, eine sukzessive durch ein multidisziplinäres Expertenteam erweiterte medizinische Referenzterminologie in logischer Sprache mit über 400.000 logischen Sätzen, wird von einer stetig wachsenden Zahl von Organsationen wie dem *Clinical Data Interchange Standards Consortium (CDISC)* oder der *US Food and Drug Administration (FDA)* verwendet. Sie beinhaltet formale Definitionen von über 100.000 Konzepten, Termen und Synonymen in der Krebsdomäne, modelliert erforschte Beziehungen zwischen über 10.000 Krebsarten und anderen Krankheiten und umfasst relevante Informationen zu 17.000 Krebsmedikationen und Kombinationstherapien.[9] In Kollaboration mit dem *Center for Biomedical Informatics Research*[10] an der *Stanford University* werden aktuell die in der hier beschriebenen Dissertation entwickelten Debuggingverfahren in das offizielle Entwicklungs- und Qualitätssicherungstool (Protégé[11]) für den NCI Thesaurus integriert.

Außerdem basiert die Realisierung des *Semantic Web* entscheidend auf der Korrektheit von (OWL) Ontologien, die Terme zur semantisch strukturierten Beschreibung von Web Ressourcen eindeutig definieren und Zusammenhänge zwischen unterschiedlichen Begriffen und Domänen präzise spezifizieren sollen.

7 Kontributionen der hier beschriebenen Dissertation

Die dieser Kurzfassung zugrunde liegende Dissertation [Ro15] umfasst im Wesentlichen folgende wissenschaftliche Beiträge:

[6] Siehe http://oor.net/

[7] Bioportal (http://bioportal.bioontology.org) ist eine große Sammlung von biomedizinischen WBn immenser Größe (mehrere Zehn- bzw. Hunderttausend logische Sätze pro WB).

[8] Siehe http://www.ihtsdo.org/snomed-ct

[9] Siehe http://www.cancer.gov/research/resources/terminology

[10] Ansprechpartner: Matthew Horridge und Tania Tudorache

[11] Siehe http://protege.stanford.edu/

(1): Diese Arbeit stellt die umfassendste und detaillierteste Behandlung der Thematik des interaktiven Debuggings (monotoner) WBn dar, die es bis dato gibt (34 Kapitel, 365 Seiten). Die Theorie wird von Grund auf eingeführt. Daher kann die Arbeit sowohl als einführende Literatur für interessierte Einsteiger gesehen werden, als auch als ausführliches Werk, das den aktuellen Stand der Forschung widerspiegelt. Diese Arbeit und die zahlreich darin enthaltenen Literaturreferenzen sollen es anderen interessierten Forschern ermöglichen, sich effizient, tief und umfassend in das Feld einzuarbeiten. Es werden darin *erstmals* formale und präzise Problemdefinitionen aller im interaktiven Debugging behandelten Probleme gegeben. Dies stellt einen ersten notwendigen und wesentlichen Schritt zur Lösung solcher Probleme und die Basis für alle weiteren Forschungsaktivitäten in diesem Gebiet dar. Weiters liefert die Arbeit Lösungsverfahren für alle formulierten Probleme, beschreibt diese auf sehr ausführliche Weise, diskutiert deren Vor- und Nachteile und beweist schließlich formal deren Korrektheit. *Derartige bewiesen korrekte, vollständige und optimale Algorithmen und die exakte Beschreibung von deren Integration zu einem voll funktionsfähigen interaktiven Debuggingsystem gab es bislang in der Literatur nicht.* Auch wurde kein anderer Teil dieser Arbeit, abgesehen von den Publikationen des Autors (mit Koautoren), zuvor in der Literatur behandelt.

(2): Die in dieser Arbeit beschriebenen Verfahren sind sehr allgemein und daher sehr vielseitig einsetzbar. Grob gesehen sind die einzigen Voraussetzungen für die Anwendbarkeit der behandelten Methoden die Monotonie der Logik, die zur Spezifikation der vorliegenden fehlerhaften WB verwendet wurde, sowie die Verfügbarkeit von geeigneten automatischen Schlussfolgerungssystemen für diese Logik. Das heißt, die dargelegten *Algorithmen sind sowohl unabhängig von der zugrundeliegenden Logik als auch unabhängig vom verwendeten Schlussfolgerungssystem.* Beispiele für kompatible Logiken finden sich auf Seite 1.

(3): Die Arbeit bietet *erstmals* eine profunde, theoretische Behandlung und Analyse von Anfrageberechnungsmethoden mit neuen Komplexitätsresultaten und einer ausführlichen Diskussion von Verbesserungsmöglichkeiten.

(4): Die Arbeit befasst sich mit der eingehenden Diskussion unterschiedlicher Möglichkeiten, Metainformationen (z.B. Log-Daten, Statistiken) in das Debugging von WBn einfließen zu lassen, die durch Extraktion von Fehlerwahrscheinlichkeiten gewinnbringend ausgenützt werden können.

(5): Es wird *erstmals* ein formaler Korrektheitsbeweis des sehr populären Algorithmus QuickXPlain (QX) [Ju04] zur Auffindung minimal-unerfüllbarer Subformeln (MUS), also von minimal fehlerhaften Teilmengen einer WB, gegeben. QX wird von zahlreichen Arbeiten im Bereich der Constraint Satisfaction Probleme, des Semantic Web und der monotonen WBn eingesetzt. Durch die demonstrierte Korrektheit von QX kann in weiterer Folge in der Dissertation *erstmals* ein vollständiger Beweis der Korrektheit, Vollständigkeit sowie Optimalität einer sehr nützlichen Best-First Variante (höchstwahrscheinliche Diagnosen zuerst) des bekannten Hitting Set Tree Algorithmus [Re87] gegeben werden.

(6): Der theoretische Zusammenhang zwischen den beiden weitverbreiteten, verwandten Konzepten des Konflikts und der Justification wird *erstmals* hergestellt. Ersteres wird

häufig im Gebiet der Modellbasierten Diagnose eingesetzt, wohingegen letzteres gebräuchlich ist im Feld der Beschreibungslogiken und des Semantic Web. Als Konsequenz dessen können empirische Resultate betreffend eines der beiden Konzepte auf das jeweilige andere entsprechend übertragen werden. Nachdem zum Beispiel jeder minimale Konflikt eine Teilmenge einer Justification sein muss und da es einen effizienten Polynomialzeitalgorithmus zur Extraktion eines minimalen Konflikts aus einer Obermenge dessen gibt, impliziert die demonstrierte Effizienz der Justificationberechnung für eine Klasse von WBn auch die Effizienz der Konfliktberechnung für diese Klasse.

(7): Zwei neue Algorithmen für die iterative Berechnung von Lösungskandidaten (führende Diagnosen) im interaktiven Debugging werden vorgestellt. Ersterer garantiert die stetige „Konvergenz" zur exakten und optimalen Lösung des interaktiven WB Debugging Problems durch die zwingende Reduktion der Anzahl der verbleibenden Lösungskandidaten nach Beantwortung jeder beliebigen beantworteten Systemanfrage an den Benutzer. Letzterer Algorithmus ermöglicht sehr mächtige Suchraumbeschneidungstechniken weswegen von diesem, speziell für große und harte Diagnoseprobleminstanzen, ein signifikant zeit- und speichersparenderes Verhalten erwartet werden kann als es existierende Algorithmen aufweisen.

(8): Die Arbeit schlägt unterschiedliche Methoden zur Selektion einer „optimalen" Anfrage an den Benutzer vor und stellt ausführliche Analysen dieser Methoden bereit. Umfangreiche durchgeführte Experimente mit WBn der realen Welt belegen, dass informierte (auf a-priori Zusatzinformationen beruhende) Auswahl der Anfrage eine *durchschnittliche Reduktion von 45% des Debuggingaufwandes* gegenüber einer nicht-informierten Greedystrategie impliziert. Außerdem wird Evidenz dafür geboten, dass sowohl die informationstheoretische als auch die Greedyvariante eine signifikant bessere Performance mit sich bringen als eine zufällige Wahl der Anfragen. Für eine neu entwickelte, auf Heuristiken basierende Methode zur Selektion der informationstheoretisch optimalen Anfrage kann gegenüber dem bestehenden Verfahren *für alle untersuchten WBn eine durchschnittliche Ersparnis der halben insgesamten Debuggingzeit* erzielt werden.

(9): Es wird ein konfigurierbares Machine Learning Verfahren (RIO) entwickelt, wodurch das Risiko eines extremen Mehraufwands von bis zu einem Faktor von 23(!), gemessen für eine Testsuite von Real-World WBn, welcher aus gegebener irreführender Fehlerinformation resultieren kann, minimiert werden kann. Dies bedeutet, dass *mehr als 95% des nötigen Aufwandes des mit dem Debuggingsystem interagierenden Benutzers gespart* werden können. Zudem wird gezeigt, dass RIO in allen getesteten Fällen, gegeben das Debugging Problem ist genügend schwer zu lösen, mindestens eine gleich gute Performance hinsichtlich Benutzeraufwand manifestiert wie alle bestehenden Verfahren. Zudem wird in den Experimenten ersichtlich, dass RIO sogar *in über 70% dieser Fälle eine echt bessere Performance als alle existierenden Verfahren* erreicht.

(10): Um die Problematik des rasch explodierenden Suchraums von Diagnoseprobleminstanzen mit minimalen Diagnosen überdurchschnittlich hoher Kardinalität zu adressieren, werden neue Mechanismen eingeführt, die auf „direkte" Art eine Menge von führenden Diagnosen errechnen können, also ohne den Umweg über Konflikte zu gehen. Obwohl dadurch im Gegensatz zum „indirekten" Standardverfahren [Re87] gewisse Optimalitäts-

eigenschaften der vom System gelieferten Ausgabe nicht mehr garantiert werden können, erweist sich die direkte Methode gerade für solche DPIs, die z.B. als Ergebnis (teilweiser) Generierung von WBn durch (fehleranfällige) automatische Systeme entstehen, als extrem wertvolles Hilfsmittel. So *ermöglicht der interaktive direkte Ansatz für diese DPIs die Berechnung von einer bzw. 30 minimalen Diagnosen in durchschnittlich neun Sekunden bzw. weniger als zwei Minuten, wohingegen die Standardmethode nicht einmal eine einzige Diagnose innerhalb eines Timeouts von zwei Stunden lokalisieren kann.* In den Fällen, in denen beide Methoden innerhalb des Timeouts zu einer Lösung kommen, unterscheidet sich die Performanz der Methoden nur vernachlässigbar.

Literaturverzeichnis

[CP71] Ceraso, John; Provitera, Angela: Sources of error in syllogistic reasoning. Cognitive Psychology, 2(4):400–410, 1971.

[Ju04] Junker, Ulrich: QUICKXPLAIN: Preferred Explanations and Relaxations for Over-Constrained Problems. In (McGuinness, Deborah L.; Ferguson, George, Hrsg.): Proc. of the 19th National Conference on AI, Sixteenth Conference on Innovative Applications of AI. Jgg. 3. AAAI Press / The MIT Press, S. 167–172, 2004.

[Me11] Meilicke, Christian: Alignment Incoherence in Ontology Matching. Dissertation, Universität Mannheim, 2011.

[Re87] Reiter, Raymond: A Theory of Diagnosis from First Principles. Artificial Intelligence, 32(1):57–95, 1987.

[Ro15] Rodler, Patrick: Interactive Debugging of Knowledge Bases. Dissertation, Alpen-Adria Universität Klagenfurt, 2015.

[St08] Stuckenschmidt, Heiner: Debugging OWL Ontologies - A Reality Check. In (Garcia-Castro, et al., Hrsg.): Proc. of the 6th International Workshop on Evaluation of Ontology-based Tools and the Semantic Web Service Challenge. Tenerife, Spain, S. 1–12, 2008.

Patrick Rodler wurde am 7. Juni 1984 in Klagenfurt geboren. Er schloss im Jahr 2002 das Gymnasium mit Bestnoten in allen Fächern ab. Danach absolvierte er die Studien der Technischen Mathematik und der Informatik sowie das Doktorat der Technischen Wissenschaften im Bereich Informatik an der Alpen-Adria Univeristät Klagenfurt (kurz AAU) allesamt mit Bestnoten in allen Fächern. Währenddessen war er seit 2010 als Forscher und seit 2012 zusätzlich als Lecturer am Institut für Angewandte Informatik (Forschungsgruppe Intelligente Systeme und Wirtschaftsinformatik) an der AAU u.a. in den Fächern Knowledge Engineering, Algorithmen und Datenstrukturen, Logik, Logische Programmierung sowie Uncertain Knowledge tätig. Seit Anfang 2016 ist er Postdoctoral Researcher in genannter Forschungsgruppe. In der Forschung befasst sich Patrick Rodler mit modellbasierter Diagnose, wissensbasierten Systemen, Wissensrepräsentationssprachen, Beschreibungslogiken sowie dem Semantic Web. In seiner Dissertation beschäftigte er sich eingehend mit dem Thema des interaktiven Debuggings von Wissensbasen und konnte darin u.a. erstmals beweisbar korrekte und optimale Verfahren zur Lösung von verschiedenen Debuggingproblemen entwickeln.