G. Görz, U. Schmid, J. Schneeberger, T. Braun (Hrsg.)
Handbuch der Künstlichen Intelligenz
De Gruyter Studium

G. Görz, U. Schmid, J. Schneeberger, T. Braun
(Hrsg.)

Handbuch der Künstlichen Intelligenz

6. Auflage

DE GRUYTER

ISBN 978-3-11-021808-4
e-ISBN (PDF) 978-3-11-021809-1
ISSN 0179-0986

Library of Congress Cataloging-in-Publication Data
A CIP catalog record for this book has been applied for at the Library of Congress.

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über
http://dnb.dnb.de abrufbar.

Druck und Bindung: Druckerei XYZ
♾ Gedruckt auf säurefreiem Papier
Printed in Germany

www.degruyter.com

1 Assistenzsysteme

Matthias Kraus, Bernd Ludwig, Wolfgang Minker, Nicolas Wagner

Dieses Kapitel des Handbuchs führt in das Thema *Assistenzsysteme* aus Sicht der Künstlichen Intelligenz ein. Zu Beginn wollen wir den Begriff des Assistenzsystems aus Konzepten der Agententheorie, aus der Nutzerperspektive und Modellen für interaktive Systeme ableiten. Während wir dies in 1.1 sehr abstrakt durchführen, indem wir Charakterisierungen aus bekannten Beschreibungen des Begriffs analysieren, nähern wir uns in 1.2 anhand eines Interaktionsbeispiels aus einer typischen Domäne und anhand eines implementierten Systems für eine weitere Domäne dem Begriff weiter an. In 1.3 tragen wir die Erkenntnisse zusammen, um das Konzept *Assistenz* zu operationalisieren. Wir nehmen dabei eine nutzerzentrierte Perspektive ein und sehen Assistenz ganz allgemein als eine geeignete Systemreaktion auf einen vom Nutzer geäußerten Bedarf bei der Lösung komplexer Aufgaben.

In 1.4 leiten wir aus der Definition Anforderungen an die Wissensrepräsentation für Assistenzsysteme ab und erläutern in 1.5, dass die Entwicklung von Assistenzsysteme auch eine Herausforderung an das Software-Engineering und nicht nur an das Reasoning darstellt. Es ergibt, dass die umfassende Analyse der Domäne und der darin relevanten Abläufe eine zentrale Aufgabe ist.

In 1.6 gehen wir auf die Operationalisierung eines für Assistenzsysteme zentralen Begriffs ein, nämlich der *Kooperation*. Wir erörtern, wie Kooperation sowohl in der Anwendungsdomäne selbst als auch für den Informationsaustausch und Koordinationsbedarf effektiv implementiert werden kann, wenn die Kommunikation vorrangig über natürliche Sprache stattfindet: welche Modelle sind geeignet, um die relevanten Abläufe abzubilden, und welchen Limitationen unterliegen sie? Die Implementierung wird in Abschnitt 1.7 vertieft: welche KI-Algorithmen sind dazu vonnöten? Wie sind sie zu parametrisieren? Wie können aktuelle Beobachtungen integriert werden (Abschnitt 1.8)?

Abschließend erörtern wir in 1.9 zentrale ungelöste Fragen für eine generelle algorithmische Theorie für Dialogverstehen und Assistenz.

1.1 Einordnung des Gebiets in die Künstliche Intelligenz

Der Begriff *Assistenz* ist in der Künstlichen Intelligenz schwer zu fassen. Im Positionspapier[1] "Künstliche Intelligenz Wirtschaftliche Bedeutung, gesellschaftliche Herausforderungen, menschliche Verantwortung" des BitKom e.V. von 2017 wird

1 siehe https://www.dfki.de/fileadmin/user_upload/import/9744_171012-KI-Gipfelpapier-online.pdf, zuletzt abgerufen am 26.09.2019

https://doi.org/10.1515/9783110218091-001

der Begriff *Assistenzsystem* im Abschnitt 3.3.4 als Ausprägung von "Intelligenz-Verstärkung" und "intelligenter Entscheidungsunterstützung" erwähnt und später an einigen Anwendungsszenarien veranschaulicht, ohne eine allgemeine Erklärung oder Definition zu geben. Interessanterweise enthält die Brockhaus-Enzyklopädie[2] gar keinen Eintrag für diesen Begriff, wohl aber für *Sprachassistent*[3]:

> [...] Software, die zwei wesentliche Elemente vereint: eine Schnittstelle zwischen Mensch und Computer auf Basis von Spracherkennung und Sprachausgabe (Sprachsynthese) einerseits sowie die Erledigung von Aufgaben andrerseits. Letzteres umfasst die Suche nach und das Zusammenstellen von Informationen, das Starten und Steuern von Programmen sowie die Ausgabe von Daten. Damit bieten Sprachassistenten einen intuitiven, bequemen Umgang mit Computern. Sie sind Teil von Cognitive-Computing-Systemen. [...]

Entscheidend sind dabei die Formulierungen *Erledigung von Aufgaben* und *Cognitive-Computing-Systemen*. Unter *Cognitive Computing* versteht Brockhaus[4]

> Rechnerverfahren, bei dem Konzepte und Techniken aus der künstlichen Intelligenz genutzt werden, um Funktionen des menschlichen Gehirns nachzuahmen mit dem Ziel, den Nutzer bei Entscheidungsprozessen (v.a. in der Wirtschaft) zu unterstützen. Wesentliche Elemente von Cognitive-Computing-Systemen sind Spracherkennung, Mustererkennung, Lernfähigkeit (z.B. auf Basis neuronaler Netze), sehr schnelle Computer und große Datenmengen (Big Data). Sie stellen die moderne Variante von Assistenzsystemen dar, über die in der künstlichen Intelligenz schon länger diskutiert wird.

Auch hier wird der Begriff *Assistenzsystem* wieder ohne weitere Erläuterung vorausgesetzt. Klar scheint vor allem zu sein, dass für die Realisierung von Assistenzsystemen KI-Verfahren benötigt werden, dass natürlichsprachliche Interfaces die zentrale Modalität für Mensch-Computer-Interaktion mit Assistenzsystemen sind, und dass sie Aufgaben erledigen und Unterstützung bei Entscheidungen bieten. Dies legt die Überlegung nahe, ob Assistenzsysteme eine eigenständige Gattung in der Künstlichen Intelligenz sind wie auch *Recommender Systems* oder *Decision Support Systems*. Der Brockhaus-Artikel zu Sprachassistenten äußert sich dazu folgendermaßen:

> Textbasierte Assistenzsysteme, oft als Chatbots bezeichnet, existieren schon länger (seit Beginn der Erforschung der künstlichen Intelligenz); Systeme, die den Nutzer, nachdem dieser auf die eine oder andere Weise seine Eingabe vorgenommen hat, unterstützen, indem sie eigenständig Aufgaben erledigen und die Ergebnisse präsentieren, werden auch Software-Agenten genannt.

2 https://www.brockhaus.de

3 siehe https://brockhaus.de/ecs/enzy/article/sprachassistent-informatik, zuletzt abgerufen am 26.09.2019

4 https://brockhaus.de/ecs/enzy/article/cognitive-computing, zuletzt abgerufen am 26.09.2019

Hier kommen zwei neue Aspekte hinzu: zunächst, dass Nutzer Eingaben vornehmen, Assistenzsysteme also interaktiv sind. Zweitens die Eigenständigkeit bei der Erledigung von Aufgaben. Diese Charakterisierung führt in eine spezielle Richtung, die eine vollständige Delegierung der Erledigung von Aufgaben von Nutzern an automatische Systeme impliziert. Sie als konstitutiv für *Assistenzsysteme* zu verstehen, würde bedeuten, dass prinzipiell fast jede Form von Software als Assistenzsystem verstanden werden kann - was einer gewissen Berechtigung ja auch nicht entbehrt, aber auch im Kontrast steht zur Definition von [46], der autonome, lernende und kooperative Agenten unterscheidet und damit einen der Eigenständigkeit entgegenstehenden Aspekt ins Spiel bringt: Kooperation. Insbesondere definiert er *smart agents* als Agenten, die zugleich autonom, lernfähig und kooperativ sind. Offen bleibt an dieser Definition, ob bestimmte Klassen von Algorithmen dafür konstitutiv sind, ein Softwaresystem als Agent zu verstehen, was ja die Definition von Cognitive-Computing-Systemen nahelegt. Sind also Assistenzsysteme autonome, lernende und kooperative Agenten, die Aufgaben unter Nutzung von Konzepten und Techniken aus der Künstlichen Intelligenz erledigen? Sinnvoll wäre diese Definition dann, wenn sich aus der Natur der zu erledigenden Aufgaben und den Eigenschaften der Autonomie, Lern- und Kooperationsfähigkeit die Verwendung von KI-Verfahren (zwingend) ergäbe, um Nutzern einen Vorteil bei ihrem Einsatz zu verschaffen, der ohne KI-Verfahren nicht erzielbar wäre.

Autonomie ist dabei die Eigenschaft eines Agenten, in seiner Domäne definierte und in der aktuellen Situation relevante Aktivitäten mit eigenen Mitteln (und ggf. aus eigenem Antrieb) durchzuführen [19]. Viele Fahrer-Unterstützungs-Systeme in KFZ sind autonom: beispielsweise funktioniert ABS ohne Zutun von Fahrern, alleine durch Auswertung von Messdaten (daher situationsrelevant) durch wohldefinierte Aktivitäten (Steuerung der Bremsen der einzelnen Räder nach einem festgelegten Reaktionsschema). Wir möchten hier festhalten, dass für eine derartige Assistenz ihr Ablauf vorgegeben und nur hinsichtlich einiger Parameter variabel ist. Es muss nicht erst ein Lösungsweg gesucht werden (vgl. *Planung* [22]). Wir müssen also Funktionsausführung (wie beim ABS) von Problemlösung (der Nutzer gibt an, welche Ziel er erreichen möchte, der kooperative Agent findet einen Lösungsweg dafür) unterscheiden. Wir werden an den Beispielen später sehen, dass die Suche nach einem Lösungsweg oft eine wichtige Aufgabe für situationsangepasste und personalisierte Assistenz ist.

Lernfähigkeit hätte ein Agent, wenn er bestimmte Parameter von Aktivitäten aus Daten optimal an Zielbedingungen anpasste, oder sogar Entscheidungen, welche Aktivitäten wann auszuführen sind, von durch Lernverfahren festgelegten Parameterwerten abhängig machte (vgl. [56, Abschnitt 2.4.6]). Hierfür sind Verfahren für Maschinelles Lernen erforderlich.

Kooperation ist charakteristisch für Agenten, die Problemlösungen für ein Ziel ermitteln, das mit anderen Agenten abgestimmt ist (sog. *gemeinsames Ziel*, siehe [76, Abschnitt 8]). Problemlösungen sind dabei Folgen von Aktivitäten (*Pläne*), die in der

aktuellen Situation beginnend nacheinander ausgeführt werden und eine Situation herstellen können, in denen ein gemeinsames Ziel erreicht ist. Ein weiteres Merkmal von Kooperation ist, dass auch eine Problemlösung mit anderen Agenten abgestimmt sein muss, sonst würden die Agenten nicht im eigentlichen Sinn kooperieren, sondern unabhängig voneinander, also jeder für sich *autonom*, versuchen, ein gemeinsames Ziel zu erreichen [76, Abschnitt 8.6]. Stattdessen müssen alle in die Kooperation involvierten Agenten dieselbe Aktionsfolge synchronisiert durchführen [76, Abschnitt 8.4]. In diesem Sinn kann Assistenz schon darin bestehen, eine Problemlösung für ein gemeinsames Ziel zu ermitteln. Dabei hat das Ziel ein Agent definiert, und ein anderer Agent (das *Assistenzsystem*) bestimmt dann eine geeignete Problemlösung, die er den anderen Agenten mitteilt. Umfassender kann Assistenz auch darin bestehen, das gemeinsame Ziel interaktiv zu definieren.

Soll ein Agent also kooperativ (anders als beispielsweise ein autonomes Anti-Blockier-System) Assistenz leisten, ergibt sich als Grundannahme ein ungleichwertiges Verhältnis zwischen den an der Kooperation beteiligten Agenten. Ein Nutzer, technisch als eine spezielle Klasse von Agent verstanden, hat die Initiative, Ziele vorzugeben, wie es für Softwareanwender ja auch typisch ist. Dagegen ist ein Assistenzsystem ein kooperierender Agent, der Ziele von Nutzern akzeptiert und Problemlösungen dafür entwickelt. Er kann einzelne Aktivitäten davon auch autonom durchführen, andere Aktivitäten führt der Nutzer durch.

Kooperation kann aber noch weitergehen: Um den Fortschritt einer Problemlösung erkennen zu können, muss der assistierende Agent die Durchführung von Aktivitäten des Nutzers und auch von eigenen beobachten können. Dazu sind wiederum spezielle KI-Verfahren erforderlich [56, Abschnitt 11, 24]. Zweck der Beobachtung ist, festzustellen, ob Aktivitäten korrekt durchgeführt wurden. Dazu muss der Agent über Wissen verfügen, welche Beobachtungen zu erwarten sind. Macht er andere Beobachtungen, liegt ein Problem vor. Hat dabei ein Nutzer eine Aktivität nicht wie erwartet ausgeführt, besteht deswegen eventuell Unterstützungsbedarf. Er kann vielfältig sein, und unterschiedliche Strategien können ihm abhelfen, die natürlich durch die Domäne der Problemlösung bestimmt sind. Den Unterstützungsbedarf zu erkennen, ist oft keine triviale Aufgabe, da ihn der Nutzer normalerweise nicht artikulieren kann, wenn ihm nicht bewusst ist, dass eine Aktivität anders als erwartet ausgeführt wurde, oder wenn er dies zwar bemerkt, er aber den Grund dafür nicht erklären kann.

1.2 Assistenzbedarf in Beispielen

Nun haben wir also einige zentrale Begriffe der Künstlichen Intelligenz besprochen, die sich aus einem genauen Blick auf die zitierten Lexikonartikel ergeben. Es hat sich dabei gezeigt, dass Assistenzsysteme verschiedene Teilbereiche der Künstlichen Intelligenz tangieren, so ähnlich wie auch die Robotik, so dass sie wohl mit Recht als eigenständige

Gattung in der Künstlichen Intelligenz angesehen werden dürfen. Diese Beobachtung hilft uns aber nicht dabei, den Begriff des *Assistenzsystems* genauer zu bestimmen. Deswegen soll – um eine Definition zu erarbeiten – zunächst an zwei Beispielen herausgestellt werden, wie Assistenz als interaktive Kooperation autonomer und lernfähiger Agenten mit Nutzern realisiert werden kann. Mit diesen Beispielen wollen wir dann eine Taxonomie von Assistenzkonzepten und Handlungsphasen begründen, die später Basis für generische algorithmische Konzepte für Assistenz(systeme) sein wird.

1.2.1 Anwendungsbeispiel 1: *Interaktion mit einem Küchenhelfer*

Das erste Beispiel stammt aus einem Corpus von in-situ-Experimenten [21] mit jeweils einer Testperson und einem Experten: Testpersonen hatten die Aufgabe, aus verfügbaren Zutaten ein Rezept ihrer Wahl zu kochen, das sie sich zuvor aussuchen sollten. Dabei half ihnen ein Experte (im Folgenden Agent genannt), der alle Fragen zur Rezeptsuche und Zubereitung von Rezepten beantworten konnte, aber nicht selbst, auch nicht unterstützend, tätig wurde. Die Experimente fanden in einer ganz normalen Küche statt. Der folgende Ausschnitt stammt aus einem dieser Dialoge. Die Transkription ist mit einer Annotation für den Assistenzbedarf, der den Nutzer TP dazu veranlasst, für das Erreichen seines Ziels (Kochen eines Gerichts) vom kooperierenden Agenten Assistenz anzufordern. Desweiteren ist jede Äußerung mit einer Hypothese annotiert, welche kommunikative Absicht TP mit der Äußerung verbindet – aus der wiederum eine bestimmte Erwartung über die Reaktion des Agenten abgeleitet werden kann. Schließlich ist die kommunikative Strategie annotiert, mit der TP dem Agenten Bedarf und Absicht bzw. Erwartung an seine Reaktion mitteilt:

ID	**Äußerung**	**Assistenz (-Bedarf) bei**	**Zweck der Interaktion**	**Strategie**
TP_1	Äh (.) Nenn mir ein Rezept mit Reis (,) Sahne und (.) Butter (?)	Entscheidung über Auswahl einer Aktion	Anforderung von Optionen	Verbalisierung des Bedarfs
A_2	Ok Also bei Chefkoch (,)(.) hab ich gesucht nach Paprika-Sahne-Reis-Rezepten (,) und gefunden (.)	Integration von Information	Erklärung von Systemausgaben	Verbalisierung der Erklärung

	ähm Rahm-Paprika-Hähnchenbrust mit Reis Paprika-Sahne-Geschnetzeltes von der Pute mit Reis gefüllte Paprika mit Hackfleisch und Reis in Tomatenso-Tomaten-Sahne-Soße (,) (.) äh fröhliche gefüllte Paprika in einer Käse-Sahne-Soße (.) ähm Paprikareis (.) ähm (.) Paprika-Sahne-Schnitzel mit Bohnenbündchen und Reis (,) (.)		Information über ausgewählte Optionen	Verbalisierung der Optionen
TP_3	Ok	Effektkontrolle	Rückmeldung zu Assistenz	Verbales Feedback
	da hab ich ja die Hälfte nicht davon Hab ja kein Fleisch		Bereitstellung von Erklärungen	Verbalisierung der Erklärungen
	Öh (.) Ähm (?) Ja Ok	Effektkontrolle	Rückmeldung zu Assistenz	Zögern
	dann brauch ich jetzt ein Rezept mit Tomaten (,) Zwiebeln (.) Paprika (.) Sahne (,) Reis (.)	Entscheidung über Auswahl einer Aktion	Anforderung von Optionen	Verbalisierung des Bedarfs
A_4	Ich hab (.) Couscouspfanne mit Paprika und Tomate (,)(.) gefüllte Paprika mit Hackfleisch und Reis in Tomaten-Sahne-Soße (,)(.) Paprika-Tomaten-Fischtopf (,) Erdnussoße mit Reis und Gemüse (,)	Integration von Information	Information über ausgewählte Optionen	Verbalisierung der Optionen
TP_5	Ich glaub so komm ich nicht weiter (lacht)	Effektkontrolle	Grad der Zielerreichung bewerten	Verbalisierung
	Ok dann mach ich es anders	Motiv- und Zielbildung	Zielwechsel	Verbalisierung
	Ähm Ich brauch ne ok also	Effektkontrolle	Rückmeldung zu Assistenz	Zögern
	ein Rezept für eine äh Paprikatomatensoße	Entscheidung über Auswahl einer Aktion	Anforderung von Optionen	Verbalisierung des Bedarfs
A_6	Ok		Verbales Feedback	
TP_7	Tomatensoße mit Paprika oder irgendwie so Paprika-Tomatensoße	Entscheidung über Auswahl einer Aktion	Anforderung von Optionen	Verbalisierung des Bedarfs

A_8	Ähm ich hab gefüllte Paprika mit Tomatensoße und Reis (.) Ähm Gefüllte Paprika in Tomatensoße gefüllte Paprika in Tomatensoße (,) (.) ähm (.)	Integration von Information	Information über ausgewählte Optionen	Verbalisierung der Optionen
TP_9	Ähm (.)	Effektkontrolle	Rückmeldung zu Assistenz	Zögern
A_{10}	Nudelauflauf Hähnchenfleisch und Gemüse mit einer Paprikatomatensoße	Integration von Information	Information über ausgewählte Optionen	Verbalisierung der Optionen
TP_{11}	Dann (.) Sag mir mal die Anweisungen zu der Paprika zu der ersten Soße	Ausführung von Aktionen	Vorschlag einer Option	Verbalisierung des Bedarfs
A_{12}	Zu gefüllter Paprika mit Tomatensoße und Reis (?)	Berücksichtigung der aktuellen Situation		Alignment
TP_{13}	Ja			Feedback
A_{14}	Ok Ähm (?) Zutaten Hackfleisch Paprika Brötchen Eier Zwiebeln Salz und Pfeffer	Integration von Information	Bereitstellung von Erklärungen	Verbalisierung von Erklärungen
TP_{15}	Toll (lacht)	Effektkontrolle	Grad der Zielerreichung bewerten	Verbalisierung
A_{16}	Und Zubereitung Zuerst das Brötchen in Wasser einlegen (,)(.) wie bei Frikadellen (,) dann die Paprika waschen und oben abschneiden sodass ein Deckel entsteht (,) den grünen Stiel dran lassen (?) Nun die Paprika von innen waschen (,) die weißen Stellen vorsichtig raus schneiden (,) und die Paprika griffbereit hinstellen	Integration von Information	Bereitstellung von Erklärungen	Verbalisierung von Erklärungen

Oberstes Ziel von Assistenz sollte ja immer sein, einer anderen Person bei der Durchführung einer Handlung zu helfen. Dies ist geschafft, wenn die Handlung erfolgreich abgeschlossen worden ist. Für die Implementierung von Assistenz ist es also unerlässlich zu wissen, welche Handlungen (einer Anwendungsdomäne) welche Ergebnisse erwarten lassen. In diesem Sinn ist A_2 im Beispiel oben die Mitteilung über eine erwartungsgemäße Durchführung der in TP_1 formulierten Handlungsanweisung. Um die Antwort A_2 geben zu können, hat der Agent ein Problem gelöst: er hat einen (jetzt noch nicht näher interessierenden) Algorithmus ausgeführt, um zu ermitteln, wie der erkannte Bedarf von TP befriedigt werden kann. Das Ergebnis des Algorithmus wird dann in A_2 formuliert und kommuniziert. Entscheidend für Assistenz ist also

die Kompetenz zur Problemlösung, die im einfachsten Fall in der Ausführung eines geeigneten Algorithmus besteht. Wie komplex kann so ein Algorithmus werden? Welche Eigenschaften muss er aufweisen? An unserem Beispiel wird schon deutlich, dass die Problemlösung nicht immer sofort in die Tat umgesetzt werden darf: Das bloße Nennen von Gerichten ist ja noch kein Kochen, sondern nur Interaktion mit Nutzern über mögliche Optionen, was in der (unmittelbar bevorstehenden) Zukunft geschehen soll, um die Aufgabe (d.h. das Problem) des Nutzers (Kochen eines Gerichts) zu lösen. Assistenz ist also zunächst rein informativ. Daher benötigen wir für die Implementierung von Assistenz einen Ansatz, der zunächst mögliche Lösungen ermittelt und erst später ausführt, wenn nicht nur das gemeinsame Ziel, sondern auch seine Problemlösung zwischen den involvierten Agenten abgestimmt sind, wie oben schon für Kooperation gefordert [38]. Algorithmen, die mögliche Problemlösungen berechnen können, sind in der Künstlichen Intelligenz als Planungsverfahren bekannt [22]. Sie liefern Lösungen „als Daten“, die dann an den Nutzer kommuniziert werden können, wie in A_2 zu sehen. Diese Eigenschaft ist zentral, weil ja der Nutzer über das Gericht entscheiden soll, das gekocht wird, nicht ein assistierender Agent. Als Konsequenz aus der Forderung nach Interaktion und Kooperation, die einen Agenten zum Assistenten machen, kann Assistenz also nicht allein durch Funktionsausführung geleistet werden [8]. Noch deutlicher wird dies an der Äußerung TP_{11}: Hier wird Information über eine Problemlösung angefordert – ein Plan, wie die ausgewählte Paprikasoße zubereitet wird. Während bei einem Kochrezept der Plan zur Zubereitung des Gerichts bis auf minimale Variationen feststeht, muss dies bei anderen Anwendungsdomänen – wie wir im nächsten Beispiel unten zeigen werden – nicht der Fall sein. Daher ist *Planen* aber nicht die einzige *kognitive Kompetenz*, die für Assistenz erforderlich ist [37]. Zur Beantwortung der Äußerung TP_1 ist die Fähigkeit, Optionen unter Berücksichtigung von situativen und nutzerspezifischen Randbedingungen ermitteln zu können, gefragt: Analog zum Planen, das mögliche Aktionsfolgen in die Zukunft vorausprojiziert, ist für Assistenz das „probeweise“ Entscheiden anhand von zu erwartenden Kriterien eines Nutzers typisch. Die Antwort A_2 sollte idealerweise Optionen für Gerichte präsentieren, die der Nutzer auch gerne essen möchte, die er zu kochen beherrscht, und die bestimmte Randbedingungen, wie die verfügbare Zeit für das Kochen, einhalten. Daher muss jede Implementierung von Assistenz auch mit unsicherem Wissen operieren können, weil für viele der eben genannten Parameter explizite Werte nie zur Verfügung stehen können. Neben den Planungsverfahren spielen also auch Klassifikations-, Optimierungs-, Empfehlungs- oder Entscheidungsunterstützungs-Verfahren eine wichtige Rolle.

Für ein allgemeines, domänenunabhängiges Konzept von Assistenz sind – als Fazit aus der bisherigen Diskussion – verschiedene Verfahren der Künstlichen Intelligenz ein Schlüssel zur Lösung. Die Frage von oben, wie komplex ein Algorithmus sein kann, lässt sich also mit „sehr hoch“ beantworten.

Unser Dialogbeispiel zeigt aber noch weitere Dimensionen von Komplexität: Erstens haben Interaktionen während eines Assistenzvorgangs ganz unterschiedliche

Bedeutung. Während TP_1 konkret angibt, wie das gemeinsame Ziel (partiell) formuliert werden soll, verfolgt der Nutzer in TP_3 einen anderen kommunikativen Zweck: Durch die Information *Hab ja kein Fleisch* beschreibt er weitere Randbedingungen für die Umsetzbarkeit von möglichen Problemlösungen, die vom Assistenzsystem auch so verstanden werden müssen – nicht jede Nutzeräußerung ist also ein direktes Kommando zur Rezeptsuche oder -präsentation.

Zweitens korrespondieren einzelne Aktivitäten einer Problemlösung nicht immer mit genau einer Äußerung: Die Äußerung TP_{11} beispielsweise bezieht sich auf die Antwort des Agenten in A_8 und damit inhaltlich auf die dort eingeführten Entitäten (und Aktivitäten), also die Vorschläge für Soßen. Der Verlauf der Interaktion von TP_5 und TP_7 über A_8 hin zu TP_11 belegt somit, dass der Assistenzbedarf für das Ziel aus TP_5 (und damit eigentlich aus TP_1) immer noch nicht befriedigt worden ist. Er zeigt auch, dass eine Problemlösung zur Befriedigung des Bedarfs komplex ist und sich über mehrere Aktivitäten erstreckt: Das Nennen von Rezepttiteln ist nur eine Aktivität. Es gehört auch die Auflistung der Zutaten und die Erläuterung der einzelnen Zubereitungsschritte dazu. Ein Gedächtnis ist deshalb erforderlich, weil sich der Agent seine Antworten aus A_8 als Teilschritt der Problemlösung für den in T_1 eingeführten Assistenzbedarf merken muss. Insgesamt ist also erst nach der Äußerung A_{16} das Problem, ein Rezept zum Kochen auszuwählen, für den Agenten zunächst gelöst worden. Denn er kann nun Informationen zur Zubereitung geben[5].

Drittens gibt es in jedem natürlichsprachlichen Dialog Äußerungen, die sich auf Wahrnehmen, Analysieren und Verstehen von Sprache beziehen, sowie explizite Formulierungen über sprachliches Handeln. Solche Äußerungen sind in den Interaktionsablauf eingebettet und müssen korrekt erkannt werden. Beispiele dafür sind die Häsitationen in TP_3, TP_5, A_{14} oder TP_{15}. Letztlich ist die Liste der „domänenfremden“ Themen überhaupt nicht eingrenzbar, ein Nutzer kann auch seine Befindlichkeiten äußern – wie häufig bei Anrufen bei Service-Hotlines, er kann auch die Domäne „verlassen“ und über ein ganz anderes Thema reden wollen, wie dies bei Smalltalk oft der Fall ist. Ein Sprachassistent muss in der Lage sein, diesen Umstand zu erkennen und eine geeignete Dialogstrategie anzuwenden, um den Diskurs wieder auf die aktuelle Problemlösung zu lenken. Die Komplexitätsdimension ist typisch für natürlichsprachliche Dialoge; bei Interaktion in anderen Modalitäten ist sie per Design des Interface ausgeschlossen, d.h. Nutzer haben überhaupt nicht die Möglichkeit, solche Inhalte mitzuteilen.

Zieht man ein Fazit aus der Diskussion des Beispiels, zeigt sich, dass Kooperationsmechanismen und die von ihnen ausgelöste (sprachliche) Interaktion zwischen den beteiligten Agenten über den Assistenzbedarf interagieren: Ein bestimmter Bedarf

5 Im weiteren Verlauf wird der Nutzer auch dieses Rezept verwerfen, weil er ja kein Fleisch hat. Eine längere Diskussion des Vorgangs ist aber hier nicht zweckmäßig, weil keine neuen Konzepte, wie Assistenz angefordert wird, mehr deutlich werden.

an Assistenz löst eine Interaktion aus, die bezweckt, dass der Dialogpartner in einer bestimmten Weise reagiert, um den Assistenzbedarf zu befriedigen (Kooperationsziel). Um den anderen Agenten den Assistenzbedarf und das Kooperationsziel mitzuteilen, bedient sich ein Agent einer bestimmten kommunikativen Strategie [62]: im Fall ausschließlich sprachlicher Interaktion ist die Strategie immer die Wahl einer geeigneten Formulierung, bei multimodaler Interaktion gibt es entsprechend der verfügbaren Modalitäten auch andere Strategien, die je nach Bedarf und Kommunikationsziel unterschiedlich geeignet sein können.

1.2.2 Warum viele Dialogmodelle für Assistenz zu einfach sind ...

Deshalb sind einfache Modelle für natürlichsprachliche Mensch-Maschine-Interaktion zur Problemlösung nicht ausreichend, wenn Assistenz in einem komplexeren Sinn verstanden wird als Sprache als Modalität für die Befehlsübermittlung an eine Maschine zu verwenden: (Frei) verfügbare Plattformen wie https://snips.ai/, https://rasa.com/, aber auch kommerzielle Lösungen wie GOOGLEs Dialogflow[6] oder MICROSOFTs Language Understanding Intelligent Service (LUIS)[7] gehen von der Annahme aus, dass eine Äußerung immer der Aktivierung einer Funktion des Assistenzsystems entspricht.

Abb. 1.1 zeigt ein Beispiel[8] für die Anwendung dieses Prinzips: Jeder Äußerung wird eine Aktivität zugeordnet, die das System im Hintergrund dann ausführen kann. Aufgabe des Sprachinterfaces ist daher "nur", aus der Äußerung die Bezeichnungen der Funktion und ihrer Parameter zu extrahieren. Gelingt dies, wird die Funktion aktiviert, eventuelle Ergebnisse sprachlich kommuniziert und damit ein Dialogsegment abgeschlossen. Alle anderen Äußerungen der vollständigen Interaktion zwischen Nutzer und Assistent sind von diesem Segment unabhängig. Der noch relativ wenig komplexe Dialog oben zeigt aber schon Gegenbeispiele dafür, dass Assistenzsysteme immer genau ein Problem lösen und kein Gedächtnis benötigen. Auch diskurslinguistisch motivierte algorithmische Dialogmodelle wie das *Information State Update*-Modell [65] basieren auf der Idee der *Adjacency-Pairs*. Im Überblicksartikel [7] beschriebene Dialogsysteme implementieren ebenfalls auf ihre spezifischen Bedürfnisse zugeschnittene Dialogmodelle, die heuristisch motiviert sind.

6 siehe https://dialogflow.com/

7 https://azure.microsoft.com/de-de/services/cognitive-services/language-understanding-intelligent-service/

8 siehe https://rasa.com, zuletzt aufgerufen am 07.10.2019

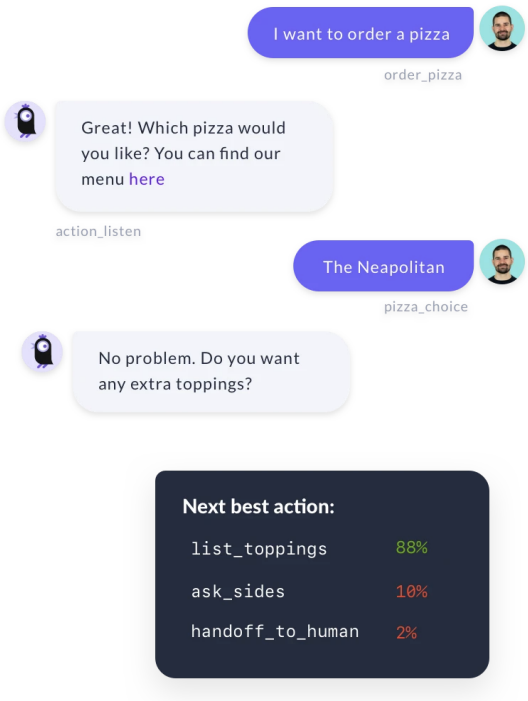

Abb. 1.1: Ein Funktionsaufruf pro Äußerung des Nutzers

1.2.3 Anwendungsbeispiel 2: *Interaktive Bedienungsanleitung*

Wir wollen an einem zweiten Beispiel verdeutlichen, um welche Komponenten und Fähigkeiten (aus dem *Cognitive Computing*) diese Modelle erweitert werden müssen. Später in 1.3 werden wir noch weitere Fähigkeiten identifizieren, die für die Verarbeitung noch komplexerer Assistenzabläufe erforderlich sind.

1.2.3.1 Fähigkeit zu planen

Dazu betrachten wir die Interaktion mit dem Heimwerker-Assistenten ROBERT [5, 31]. Das Ziel der Assistenz ist es dabei, unerfahrene Nutzer Schritt für Schritt durch Heimwerkerprojekte zu leiten und ihnen dabei die Bedienung von elektrischen Werkzeugen, wie z.B. einer Stichsäge zu beizubringen.

Für diese Form von Assistenz, bei der Lösungen für komplexe Aufgaben vorausberechnet werden sollen, ist die Fähigkeit, solche Lösungen zu planen erforderlich. Statische Dialogskripte sind dafür nicht optimal, weil die Situation, in der eine Lösung gesucht wird, bei der Definition des Skripts nicht bekannt ist. Stattdessen ist die

Information, die die Situation charakterisiert, erst zur Ausführungszeit vollständig bekannt. Daraus resultiert der Bedarf.

Wir illustrieren dies an der Anwendungsdomäne von ROBERT: Die Interaktion mit ROBERT beginnt auf Initiative eines am Heimwerken interessierten Nutzers. Zunächst präsentiert ROBERT dem Nutzer eine Liste der verfügbaren Projekte (z.B. Bau eines Schlüsselbretts, eines Vogelhauses, eines Tisches, Renovierung einer alten Tür oder eines Stuhls). Nach der Auswahl eines Projekts listet ROBERT die verfügbaren Werkzeuge und Materialien auf. Auf Basis dieser Informationen generiert ROBERT mit seinem Planer eine Abfolge von Aktionen, d.h. einen Plan, wie mit den verfügbaren Werkzeugen und Materialien das gewählte Projekt realisiert werden kann. Hier kommt der Vorteil der Planung zur Geltung: ROBERT erstellt nicht einfach eine fest programmierte Liste von Anweisungen für jedes Projekt, sondern entscheidet automatisch, welche Aktionen mit den aktuell verfügbaren Werkzeugen und Materialien durchgeführt werden können und müssen, um das Projekt erfolgreich abzuschließen. Für die Erstellung des Plans nutzt ROBERT sowohl ein Planungsmodell als auch Faktenwissen über Werkzeuge und deren Konfigurationen, das in einer Ontologie gespeichert ist [58].

1.2.3.2 Situationsadäquate Kommunikation

Eine weitere grundlegende Erkenntnis, die sich schön an ROBERT illustrieren lässt, ist, dass jeder Plan an Nutzer und Situation angepasst kommuniziert werden muss. Dabei sind angemessene Entscheidungen über Modalitäten und inhaltliche Details zu treffen.

Bei ROBERT werden dazu erstellte Pläne zur Präsentation an das Dialogsystem übergeben. Das Modul präsentiert sie in Form von Schritt-für-Schritt-Anleitungen, wobei jede Aktion einer Anweisung entspricht. Die Anweisungen setzen sich aus einer textlichen Beschreibung der Aufgabe, einem Bild sowie einem handlungsbeschreibenden Video zusammen[9]. Dabei wird das Wissen in der Ontologie von ROBERT ausgenutzt, um geeignete Medieninhalte zu finden. Diese speichert sowohl textuelle Beschreibungen der einzelnen Aktionen als auch Bilder und Videos, wie man Aktionen durchführt. ROBERT findet das am besten geeignete Lehrmaterial mittels Ontologie-Argumentation anhand der Aktionen und deren Parameter (z.B. Konstanten).

Während dem Benutzer der gesamte Plan präsentiert wird, wird die Abstraktion als eine Übersichtsleiste am oberen Bildschirmrand angezeigt. Hier wird die Position der aktuell angezeigten Aktion in der abstrakten Planabfolge angezeigt. Auf diese Weise weiß der Nutzer ungefähr, an welchem Punkt er sich im Projekt befindet und wie viel noch zu tun ist, d.h. er kann seinen Fortschritt verfolgen. Dadurch soll der Nutzer für die weitere Durchführung des Heimwerkerprojekts motiviert werden.

9 Beispielhafte Illustrationen finden sich auf https://www.uni-ulm.de/in/ki/research/projects/companion-technologie-fuer-die-heimwerkerunterstuetzung/, zuletzt abgerufen am 02.11.2019.

Diese Design-Entscheidungen resultieren bei der Entwicklung von Assistenzsystemen aus präzisen Analysen von z.B. in WoZ-Experimenten erhobenen Dialogabläufen. Sie werden oft aufgrund von Ergebnissen von Usability-Evaluationen mit Testpersonen modifiziert — die Entwicklung von Assistenzsystemen stellt also auch eine Herausforderung an das Software-Engineering (siehe dazu später den Abschnitt 1.5) dar.

1.2.3.3 Beobachtung und Diagnose

Eine dritte Erkenntnis ist, dass ein Assistenzsystem Handlungen von Nutzern beobachten muss. Eher offensichtlich ist, dass so festgestellt werden kann, ob geplante Anweisungen auch umgesetzt werden. Wichtig für Assistenz ist aber auch, dass das Assistenzsystem in der Lage sein muss zu diagnostizieren, dass es Nutzer überfordert hat: jede Anweisung eines Plans stellt (implizit) Erwartungen an Kompetenzen von Nutzern in der Anwendungsdomäne, über die sie aber nicht notwendigerweise verfügen. Nutzer benötigen deshalb manchmal — ohne dass dies planbar wäre — zusätzliche Informationen oder sogar andere Lösungsvorschläge.

Bei ROBERT ist diese Erkenntnis folgendermaßen umgesetzt: Der Nutzer kann immer nach zusätzlichen Informationen zu Handlungsanweisungen fragen. Der Assistent könnte beispielsweise den Nutzer anweisen, ein Holzsägeblatt in eine Säge einzusetzen, ohne dass dieser weiß, wie man diesen speziellen Sägeblatttyp erkennt. Hier kann ROBERT Beschreibungen von Objekten – einschließlich ihrer visuellen Merkmale – bereitstellen, die ausschließlich auf den in der Ontologie gespeicherten Informationen basieren. Zusätzlich zu den detaillierten Schritt-für-Schritt-Anleitungen generiert ROBERT auch eine Abstraktion des präsentierten Plans unter Verwendung der zugrundeliegenden Aufgabenhierarchie. Dabei wird ausgenutzt, dass die Heimwerkeraufgabe an sich aus mehreren Teilaufgaben mit Unterschritten besteht. Beispielsweise muss der Nutzer, um ein Schlüsselbrett zu bauen, zuerst ein Brett in zwei Teilbretter zersägen, was Handlungsschritte mit einer Stichsäge beeinhaltet.

1.2.3.4 Anforderungsanalyse

Im Falle von ROBERT wurde anhand einer initialen Studie festgestellt, welche Probleme Nutzer im Verlauf eines Heimwerkerprojekts erfahren, und wie sie diese äußern. Dazu ließ man unerfahrene Nutzer mit konventionellen Hilfsmitteln, z.B. Papieranleitung, DIY-Handbuch, ein Heimwerkerprojekt durchführen, wobei sie ermuntert wurden, ihr Vorgehen und ihre Probleme zu verbalisieren. Zusätzlich wurde ihnen angeboten, sich im Falle von unüberwindbaren Problemen an einen „DIY-Experten“ zu wenden. Dieser befand sich in einem getrennten Raum und konnte per Mikrofon aufgerufen werden. Diese Äußerungen wurden gesammelt und genutzt, um das Sprachverständnis des Assistenten zu modellieren.

Dieses datengetriebene Vorgehen, um eine empirische Grundlage für die Domänenanalyse zu schaffen, wurde auch in anderen Projekten zum Dialogverstehen

angewandt. In [27] wird es für das TRAINS-System beschrieben, in [32, 60] für das VERBMOBIL-Dialogsystem.

1.3 Eine Definition für Assistenzsysteme

Nachdem wir nun an mehreren Beispielen und Vergleichen mit existierenden Ansätzen und implementierten Lösungen illustriert haben, worin die Komplexität bei der Implementierung von Assistenz besteht, können wir eine Definition für den Begriff des *Assistenzsystems* geben: es ist ein Software-Agent, der sich durch (multimodale) Interaktion, Kooperation, Lernfähigkeit und Problemlösungs-Kompetenz für gemeinsame Ziele auszeichnet. Er kann einzelne Aktivitäten eines Plans autonom ausführen, ihre Effekte und auch die von Aktivitäten, die ein Nutzer ausgeführt hat, beobachten und Abweichungen zwischen ihnen und einer vom Plan vorgegebenen Erwartung vergleichen. Er kann, wenn tatsächliche und vorgegebene Effekte nicht übereinstimmen, Strategien zur Planreparatur und Neuplanung anwenden[36, 38]. Für diese Fähigkeiten verfügt er über explizit repräsentiertes Domänenwissen.

Unter einem *Task* verstehen wir ein gemeinsames Ziel von Nutzer und Assistenzsystem, zu dem das Assistenzsystem über Problemlösungs-Kompetenz verfügt.

1.3.1 Der Bedarf als zentrales Konzept

Bei der Besprechung der Beispiele wurde deutlich, dass Assistenz mindestens einen aktuell auszuführenden Task, grundlegende Kooperationsmaximen, nur Richtiges kommunizieren und nur zielführende Aktivitäten ausführen, und die Fähigkeit, Assistenzbedarf zu identifizieren, voraussetzt.

Assistenzbedarf ist immer anwendungsspezifisch, egal, ob wir Problemlösungen für Aufgaben, mögliche Diagnosen zur Ausführung von Aktivitäten oder Strategien zur Beseitigung der Diagnosen betrachten. Dennoch ist es sinnvoll, eine Systematik dafür zu entwickeln, die als Grundlage für eine Theorie der Assistenzsysteme dienen kann: Die hier benutzten Bezeichnungen für Assistenzbedarf stammen aus der Taxonomie von Wandke [67]. Sie ist vollständig in Tabelle 1.1 dargestellt. Wandke unterteilt darin jede Aktivität in sechs einzelne Phasen, in denen jeweils typische Formen von Assistenzbedarf auftreten können. Sie sind in der linken Spalte aufgeführt. Um auf den Bedarf zu reagieren, kann ein Agent Assistenz vom in der rechten Spalte aufgeführten Typ anbieten. Wie sie konkret realisiert wird, ist natürlich von der jeweiligen Domäne abhängig, in der Assistenz geleistet wird. In seiner Arbeit gibt Wandke dafür viel interessante Beispiele.

Für die Entwicklung von Assistenzsystemen bietet Wandkes Taxonomie den Vorteil, die Aktivitäten einer Anwendungsdomäne systematisch zu analysieren, um festzustellen, in welchen Phasen jeder Aktivität, die Nutzer durchführen können,

Schwierigkeiten entstehen, bei denen Nutzer potenziell Unterstützung benötigen werden. Daraus ergibt sich ein Leitfaden dafür, welches Wissen über die Anwendungsdomäne für ein Assistenzsystem modelliert werden muss.

1.3.2 Companion-Technologie: Eine Realisierung der Definition von Assistenzsystemen

Eine Charakterisierung von Assistenz durch Agenten bietet aus der Perspektive der Informatik das Konzept der *Companion-Technologie*: Wie im einleitenden Kapitel von [9] beschrieben, besteht eine Möglichkeit für die softwaretechnische Operationalisierung der oben gegebenen Definition für Assistenzsysteme darin, die Funktionalität der Anwendungsdomäne im Stil eines Application Programming Interface bereitzustellen, so dass elementare Funktionen (d.h. Aktivitäten einer Problemlösung) von einer komplementären Systemkomponente aktiviert werden können. Parallel dazu müssen Interface-Funktionalitäten in analoger Weise bereitgestellt werden. Die Funktionalität sowohl für die Anwendung selbst als auch die Interaktion mit dem Assistenzsystem wird dann mit Methoden der Wissensrepräsentation so formalisiert, dass die komplementäre Komponente, der Companion, mit den oben erläuterten KI-Verfahren Problemlösungen planen kann, deren Aktivitäten aus Funktionsaufrufen der bereitgestellten API bestehen – oder aus Aktivitäten, die Nutzer ausführen sollen. Auf diese Weise wird erreicht, dass Assistenzsysteme Problemlösungen anpassen können an variable Randbedingungen unter denen sie eingesetzt werden (siehe dazu das Beispiel oben von Robert). Insbesondere kann eine Problemlösung durch geeignete Angabe der initialen Situation und des gemeinsamen Ziels an Kompetenzen von Nutzer und technischem System adaptiert und für einzelne Nutzer personalisiert werden. Bei der Planausführung können die Kooperationsprinzipien aus Wandkes Taxonomie bei der Diagnose einer Abweichung zwischen den erwarteten und beobachteten Effekten einer Aktivität umgesetzt werden, wie es in den beiden Beispielen oben mehrfach illustriert wurde. In [9, Abschnitt 1] erläutern die Autoren, dass auf diesem Weg ein Wechselspiel zwischen kognitiven Prozessen (und deren Simulation bzw. Realisierung im Softwaresystem) der Nutzer und des Systems implementiert wird [6].

Um Effekte von Aktivitäten beobachten zu können, kommen lernfähige Verfahren zur Klassifikation von Beobachtungen der Nutzer selbst und der Umgebung, in der die Kooperation zwischen Nutzer und Assistenzsystem stattfindet, zum Einsatz [9, Abschnitt 2]. Diese Klassifikation ist erforderlich, um aus der Differenz zwischen Beobachtung und Erwartung Unterstützungsbedarf in den einzelnen Phasen der Durchführung einer Aktivität (siehe Tabelle 1.1) zu erschließen.

Motiv- und Zielbildung	
Schaffung eines optimalen Aktivierungsniveaus	**Aktivierungsassistenz**
Verstärkung eines Motivs	**Coach-Assistenz**
Hemmung eines Motivs	**Warn-Assistenz**
Anregung eines Zielwechsels	**Orientierungsassistenz**
Informationsaufnahme	
Bereitstellung von Signalen	**Anzeigefunktion**
Signalverstärkung	**Verstärkungsassistenz**
Erzeugung von Redundanz	**Wiederholungsassistenz**
Transformation von Signalen in andere Modalitäten	**Präsentationsassistenz**
Integration von Information, Berücksichtigung der aktuellen Situation	
Bereitstellung von Erklärungen	**Beschriftungen, Anleitungen, Hilfetexte**
Bereitstellung externer Bezugssysteme	**Übersetzungsassistenz**
Erklärung von Systemausgaben	**Erklärungsassistenz**
Entscheidung über Auswahl einer Aktion	
Information über alle Optionen	**Angebotsassistenz**
Information über ausgewählte Optionen	**Filterassistenz**
Vorschlag einer Option	**Beraterassistenz**
Vorschlag und Ausführung, wenn der Nutzer zustimmt	**Delegationsassistenz**
Vorschlag und Ausführung, wenn der Nutzer nicht widerspricht	**Übernahmeassistenz**
Ausführung mit Information an den Nutzer	**informierende Ausführungsassistenz**
Ausführung ohne Information an den Nutzer	**stille Ausführungsassistenz**
Aktionsausführung: Wie soll die Aktion durchgeführt werden?	
Verstärken von Aktionen	*power*-**Assistenz**
Verkürzen einer Aktionsfolge	*short cut*-**Assistenz**
Alternative Modalitäten bereitstellen	**Eingabeassistenz**
Effektkontrolle	
Auswirkungen wahrnehmbar machen	**Rückmeldungsassistenz**
Grad der Zielerreichung bewerten	**Kritikassistenz**

Tab. 1.1: Taxonomie für Assistenzfunktionen. Jede Phase einer zielorientierten Handlung benötigt eine spezielle Form der Unterstützung, die von einem Assistenzsystem angeboten werden kann.

1.3.3 Problemlöse- und State-Tracking-Kompetenzen von Assistenzsystemen

Damit erhält das Companion-Prinzip eine Begründung aus der Perspektive der Nutzung und der Nutzer. Auf diesem Prinzip aufbauend, können wir, wenn wir noch berücksichtigen, dass während der Zeit, in der Assistenz geleistet wird, im Allgemeinen mehrere Tasks parallel aktiv sein können, folgende zentrale Aufgaben für die Implementierung von Assistenzsystemen identifizieren:

- Erkennen neuer Tasks während des Assistenzvorgangs
- Ermitteln von Problemlösungen für einzelne Tasks
- Ausführung von Problemlösungen
- Bestimmen des aktuellen Fortschritts in jedem Task
- Ausführung einzelner Aktivitäten, die das Assistenzsystem autonom durchführt
- Beobachtung des Fortschritts von Aktivitäten, die Nutzer durchführen
- Feststellen der aktuellen Phase einer beobachteten Aktivität
- Vergleich der dafür erwarteten mit den beobachteten Effekten
- Diagnose des Assistenzbedarfs gemäß der Taxonomie in 1.1
- Ermittlung einer optimal geeigneten, auf die Aktivität in der Domäne spezialisierten Strategie, um Assistenz leisten zu können
- Fähigkeit zur Planreparatur bzw. Neuplanung, um das gemeinsame Ziel an die erwarteten Effekte der (Assistenz-)Strategie anpassen zu können.

Mehrere parallel aktive Tasks können wir im Dialog mit dem Küchenhelfer beobachten: In A_{12} reagiert der Küchenhelfer nicht direkt auf den Assistenzbedarf, der durch TP_{11} bestimmt wird, sondern hat eigenen Bedarf bei der Berücksichtigung der aktuellen Situation: er ist sich unsicher, auf welche in A_8 und A_{10} vorgeschlagene Option sich der Nutzer in TP_{11} bezieht. TP_{13} befriedigt den Bedarf, so dass der Küchenhelfer in A_{14} mit der Ausführung der Problemlösung zur Präsentation eines Rezepts fortfahren kann.

Dieser Abschnitt des Dialogs zeigt auch, wie wichtig es ist, neue Äußerungen dem richtigen Task zuzuordnen: Für A_{12} hat der Küchenhelfer schon die Hypothese ermittelt, dass die in TP_{11} genannte Paprikasoße zu den Suchergebnissen gehört, die der Küchenhelfer gefunden hat, als er die entsprechende Aktivität zur Lösung des in TP_5 und TP_7 definierten Tasks *Finden eines Rezepts unter vorgegebenen Randbedingungen* ausführte. Diese Hypothese kann nur dann aufgestellt werden, wenn zur Problemlösung des Tasks auch die Teillösung, ein Rezept zu präsentieren, gehört. An dieser Stelle offenbart sich der enge Zusammenhang zwischen dem kontextabhängigen Verstehen von Äußerungen und dem im Assistenzsystem repräsentierten Domänenwissen, das mit der Domänenfunktionalität, zu der assistiert wird, korrespondieren muss. Könnte der Küchenhelfer die Rezeptschritte nicht nur präsentieren, sondern zusammen mit dem Nutzer ausführen, würde er vermutlich zuerst fragen, welche Absicht der Nutzer tatsächlich hat. Könnte er die Rezeptschritte nicht einmal

präsentieren, sondern nur ihre Titel, würde er TP_{11} nicht als Fortsetzung eines Tasks, sondern als implizit Beginn eines neuen erkennen.

Anders verhält es sich bei TP_5: hier muss der Küchenhelfer erkennen, dass der Nutzer den aktuellen Task für gescheitert hält. Dies erkennt der Küchenhelfer am ersten Satz in TP_5. Dass der Nutzer eine neue Suche starten möchte, wird am zweiten Satz deutlich, mit dem ein Zielwechsel formuliert wird, während der dritte Satz einen neuen Task identifiziert. TP_5 bezieht sich also auf verschiedene Phasen der aktuellen Aktivität des aktuellen Tasks, und sogar auf Phasen einer Aktivität eines neuen Tasks. Der Küchenhelfer kann daraus auch ableiten, dass beobachtete Effekte (Effektkontrolle: Ziel nicht erreicht) von den erwarteten abweichen. Seine Strategie, darauf zu reagieren, besteht in der entsprechenden Markierung des Tasks und in der Initialisierung eines neuen.

Die Problematik paralleler Tasks, auf die sich Äußerungen in nicht immer eindeutig vorhersagbarer Weise beziehen, wird in folgendem Ausschnitt aus einem anderen Dialog mit dem Küchenhelfer besonders deutlich:

ID	**Äußerung**	**Assistenz (-Bedarf) bei**	**Zweck der Interaktion**	**Strategie**
TP_1	Ok (.) Dann machen wir doch das (.) Und die Sahnesoße von dem (.) anderen Gericht was du da gerade vorgelesen hast, also das davor	Effektkontrolle	Grad der Zielerreichung bewerten	Verbalisierung der Auswahl
A_2	Von dem Schinken	Informations-aufnahme	Erzeugung von Redundanz	Rückversicherung durch Wiederholung
TP_3	Ja(.)	Effektkontrolle	Grad der Zielerreichung bewerten	Explizites Feedback
	Also wir nehmen halt dann Spargel anstatt Schinken (.)	Effektkontrolle	Grad der Zielerreichung bewerten	Explizites Feedback
	Wie viel Spa- ähm wie viel muss man nochmal vom Ende wegschneiden vom Spargel (?)	Berücksichtigung der aktuellen Situation	Anforderung von Erklärungen	Verbalisierung des Bedarfs
A_4	Ähm (-)			Zögern
TP_5	Zwei bis drei Zentimeter (?)	Berücksichtigung der aktuellen Situation	Anforderung von Erklärungen	Verbalisierung des Bedarfs
A_6	Ja ich glaube schon (.) (..) Ähm genau (.)	Berücksichtigung der aktuellen Situation	Bereitstellung von Erklärungen	Zögern

TP_7	Zwei bis drei (.)	Effektkontrolle	Grad der Zielerreichung bewerten	Explizites Feedback
A_8	Bei sehr frischem Spargel reicht es ein bis zwei Zentimeter abzuschneiden liegt der Spargel schon länger nehmt ruhig vier bis fünf weg (.)	Berücksichtigung der aktuellen Situation	Bereitstellung von Erklärungen	Verbalisierung der Information
TP_9	Also frischer Spargel (?) Ein bis zwei (;)	Effektkontrolle	Grad der Zielerreichung bewerten	Explizites Feedback
A_{10}	Und wenn er schon länger liegt vier bis fünf (.)	Berücksichtigung der aktuellen Situation	Bereitstellung von Erklärungen	Verbalisierung der Information

Hier referiert TP_1 ähnlich wie oben auf einen schon initalisierten Task. In TP_3 wird dann ein neuer Task begonnen, weil der Nutzer Hilfe dabei braucht, die Spargelenden richtig abzuschneiden. Dieser Task ist kein Schritt des Rezepts und läuft parallel mit der schrittweisen Ausführung des ausgewählten Rezepts. Die Erklärungen ziehen sich über alle weiteren Äußerungen hin. Um sie richtig zu interpretieren, muss der Küchenhelfer immer verstehen, dass sie sich nach wie vor auf denselben Task beziehen. Die Äußerungen TP_5, TP_7 und TP_9 dienen dann auch gar nicht der weiteren Ausführung eines Tasks der Anwendungsdomäne, sondern beziehen sich auf einen inhärent gegebenen Task, nämlich das Management der Interaktion durch gegenseitiges Rückversichern, dass beide Kooperationspartner von denselben Annahmen ausgehen. Wie schon im Dialog mit ROBERT zeigt sich auch hier, dass die Kontrolle von Assistenz komplex ist, und sich auf die Anwendung, die Kooperation, die Interaktion und auf die Disposition der Agenten zu allen Tasks (in den Beispielen oft am Zögern beobachtbar) erstreckt. Die Interaktion selbst kann dabei, je nach Modalität, ihre eigene Komplexität entwickeln, weil auch alle Vorgänge des Wahrnehmens von Äußerungen der Kooperationspartner und des Verstehens der Äußerungen sowie ihre Integration in den Diskurskontext Aktivitäten sind, die während einer natürlichsprachlichen Kooperation durchgeführt werden und genauso wie Aktivitäten der Domäne Unterstützungsbedarf provozieren können (z.B. weil die Wahrnehmung nicht funktioniert, die Syntax oder Semantik unklar oder mehrdeutig sein können, oder die Einbettung in den Diskurskontext scheitert). In unseren beiden Beispielen kommen derartige Phänomene nicht vor, viele Arbeiten wie [1, 40, 27, 49] thematisieren sie aber ausführlich.

Eine Integration der dort gewonnenen Erkenntnisse in das hier vorgestellte Konzept der Kooperationskontrolle durch das Verfolgen parallel aktiver Tasks kann erfolgen, wenn auch alle oben genannten Aspekte als mögliche Effekte von Kooperations-, Dialog- oder Selbstreflexions-Handlungen verstanden werden. Dies setzt dann natürlich voraus, solche Handlungen ebenfalls als Tasks zu modellieren, Unterstützungsbedarf bei einzelnen Phasen von Aktivitäten dieser Tasks zu erfassen und

Assistenzstrategien dazu zu formulieren [39]. Wir wollen später auf dieses Thema zurückkommen, nachdem wir zunächst im Folgenden besprechen, wie welches Wissen für Assistenzsysteme repräsentiert werden, und wie ein Algorithmus zur Kooperationskontrolle gestaltet werden kann.

1.4 Wissensrepräsentation für Assistenzsysteme

1.4.1 Linguistisches Wissen

Alle Dialogbeispiele zeigen exemplarisch die großen Herausforderungen, die zu bewältigen sind, damit Äußerungen richtig verstanden werden; der erste Aspekt dabei ist die Zuordnung von Äußerungen zur richtigen Domäne: bezieht sie sich auf den aktuellen Stand der Kooperation, der Interaktion zwischen Nutzer und Agent, auf die aktuelle Problemlösung, oder auf einen anderen Aspekt im Kontext des kooperativen Handelns, über den das Assistenzsystem kommunizieren kann?

Um entsprechende Inhalte von Äußerungen zu verstehen, wurden für ROBERT Konzepte aus dem Planungsmodell der Domäne (Problemlösung) in ein alle Aspekte umfassendes Sprachverständnismodell integriert.

Um Fachbegriffe – und damit die Semantik einzelner Wörter in Äußerungen – erkennen zu können, ist eine enge Kopplung mit der Wissensrepräsentation erforderlich. Deshalb wurde das domänenspezifische Faktenwissen der Ontologie genutzt, um semantische Repräsentationen von Heimwerkerkonzepten zum Sprachverständnis zu erstellen.

Die Beispiele haben auch verdeutlicht, dass Äußerungen oft den Zweck haben, unvollständige Information für die aktuelle Aktivität eines Tasks zu ergänzen oder Missverständnisse in einem bestimmten Aspekt des Handlungskontexts aufzulösen. Als Strategie für die Aktion auf derartigen Abweichungen von erwarteten Effekten führt die Dialogkomponente von ROBERT Klarifizierungsdialoge [52] mit dem Nutzer zu führen. Das Sprachverständnismodell enthält dafür Konzepte aus der Dialoghistorie, so dass Nutzeraußerungen, die etwas aufklären möchten, verstanden werden können. Beispiele aus dem zweiten Dialog mit dem Küchenhelfer sind: *von dem (.) anderen Gericht* oder *was du da gerade vorgelesen hast, also das davor* (Äußerung TP_1).

1.4.2 Wissen über Tasks

In einem zweckrationalen, zielgerichteten Dialog, wie er für Assistenzszenarien typisch ist, wird erwartet, dass jede Äußerung einen Schritt zum Erreichen des Ziels beiträgt. Während die Beispiele oben zeigen, dass es von dieser Überlegung in tatsächlichen Interaktionen Abweichungen gibt, ist sie grundsätzlich doch plausibel, weil die Interaktion den Zweck hat, das Informationsdefizit der Dialogpartner zu verringen.

Oft sind die Domänen-Tasks selbst sehr einfach, zum Beispiel Suchen nach einer Zugverbindung [44], Buchen nach Restaurants [77], Erzeugung von Suchanfragen an ein automatisiertes FAQ-System [24, 66] – es gibt viele weitere Anwendungsszenarien. Charakteristisch für alle ist, dass die Interaktion Werte für Parameter einer domänenspezifischen Funktion „einsammelt“. Die Problemlösung besteht also aus einer einzigen Aktivität. Nichtsdestotrotz ist der erwartete Interaktionsverlauf eine Konsequenz aus der Problemlösung.

Für sprachbasierte Assistenzsysteme dieses Typs besteht das Wissen über den Task also in der Kenntnis der zu erfragenden Parameterwerte. Dementsprechend haben sich zur Repräsentation des Wissens Modelle durchgesetzt, die gerade dafür geeignet sind, nämlich endliche Automaten bzw. die ihnen entsprechenden Zustandsübergangsgraphen [41]. Dort stehen stehen für bestimmte Zustände eines Tasks: für welchen Parameter hat der Nutzer nun einen Wert genannt? Kanten geben Aktivitäten an, die Zustandsübergang ermöglichen, also mögliche Äußerungen des Nutzers, die in einem Taskzustand sinnvoll sind oder in einem Dialogcorpus beobachtet wurden. Da sich Nutzer nicht immer erwartungsgemäß verhalten, besteht die Herausforderung in der Spezifikation der Zustandsübergangsrelation des Automaten: oft wird sie als Bigramm-Verteilung aus einem annotierten Corpus trainiert (wie z.B. in [24]).

Um die Sprachverstehenskomponente eines Assistenzsystems zu realisieren, die aus Nutzeräußerungen heraus den Fortschritt eines Tasks beobachten können soll, sind Bigramme von Zuständen alleine aber nicht ausreichend, weil der Nutzer den Zustand ja nicht direkt nennt. Vielmehr ist er aus Äußerungen abzuleiten, und nicht direkt beobachtbar. Daher wird die Bigramm-Verteilung ergänzt um eine bedingte Verteilung über mögliche Äußerungen in einem Zustand. Das Führen einer Interaktion wird als Markov-Prozess verstanden [53], so dass jeder Dialog als Instanz eines Hidden-Markov-Modells interpretiert werden kann. Wird ein neuer Dialog geführt, kann die Wahrscheinlichkeitsverteilung für Sequenzen von Zuständen benutzt werden, um den aufgrund des aktuellen Zustands und der aktuellen Nutzeräußerung wahrscheinlichsten neuen Zustand zu ermitteln.

Mit dieser Berechnung alleine kann ein Assistenzsystem aber keinen zielorientierten Dialog führen, da eine solche Auszeichnung der Zustände als gemeinsames Ziel ja zunächst gar nicht gegeben ist. Ergänzt man sie und quantifiziert man den Beitrag einer Äußerung für das Erreichen des Ziels (über einen Nutzen- oder *Reward*-Funktion), dann kann das Führen einer Interaktion als Markov Decision Process verstanden werden; da die Zustände, wie oben erläutert, nicht direkt beobachtbar sind, sogar als Partially Observable Markov Decision Process [74]. Dadurch wird es möglich, optimale Entscheidungen zu lernen [61], welche Aktivität das Assistenzsystem aufgrund des aktuellen Task-Zustands und der neuesten Äußerung des Nutzers als nächste durchführen soll, um das gemeinsame Ziel schnellstmöglicht zu erreichen. Lernverfahren sind in [77] beschrieben.

Dieses Modell zur Repräsentation von Wissen über Tasks ist dafür geeignet, einen Task gleichzeitig zu verfolgen. Wie unsere Beispiele oben zeigen, ist es also für

viele interessante Assistenzszenarien, in denen die Problemlösung in einem komplexen Task besteht, nicht ausdrucksstark genug. In den folgenden Abschnitten wollen wir daher das einfache Modell ausbauen, um komplexere kooperative Problemlösungen verarbeiten zu können.

1.4.3 Wissen über die Domäne

Dazu wenden wir uns zunächst wieder der Assistenzdomäne und der Frage, wie Wissen über sie repräsentiert und verarbeitet werden kann, zu. Das Beispiel zu ROBERT hat gezeigt, dass die Problemlösung für einen Task oft eine ganze Sequenz von Aktivitäten sein kann. Welche Sequenz konkret löst, ist oft abhängig vom Verlauf der Kooperation und der Situation, in der sie begonnen hat. Deswegen kommen in Companion-Systemen wie ROBERT Planer zum Einsatz. Sie bieten den Vorteil, dass sie – auf Basis einer symbolischen Beschreibung der Ausgangssituation, in der ein Task ausgeführt werden soll, und des gemeinsamen Ziels eine zielführende Sequenz von Aktivitäten ermitteln können. Dazu benötigen sie auch eine symbolische Beschreibung eben dieser Aktivitäten, d.h. eine Formalisierung ihrer Semantik: in welcher Situation ist eine Aktivität ausführbar (*Vorbedingungen*) und zu welchen kann sie führen (*Effekte*). Je nach Planungsansatz werden nur grundlegende Aktivitäten [18], zu denen dann auch eine Funktion des Assistenzsystems ausgeführt werden kann, oder Tasks und ihre Strukturierung in Zwischenziele und Subtasks [16, 17] formalisiert. Ist die Beschreibung jeder möglichen Situation (aus einer endlichen Menge von Situationen) endlich, kann Planen als Suche eines Pfads von einer Ausgangssituation zu einem gemeinsamen Ziel im Raum aller Situationen interpretiert werden. Denn unter diesen Bedingungen sind die Situationen aufzählbar. Zwei Situationen sind durch eine Aktivität miteinander verbunden, wenn die eine alle Vorbedingungen und die andere alle Effekte der Aktivität erfüllt.

Der Aufwand für die Wissensrepräsentation ist dabei recht groß: alle Funktionen des Assistenzsystems müssen in einer Planungssprache implementiert werden, der Zustand des Systems zu Beginn der Planung einer Problemlösung muss symbolisch repräsentierbar sein, Aktivitäten des Nutzers müssen wie Funktion des Systems realisiert werden. Dann sind noch die Tasks, in denen Assistenz geleistet werden kann, zu formalisieren. In der Domäne des Küchenhelfers, der auch während des Kochens assistieren kann, ist beispielsweise ein Teilziel eines Rezepts für Schweinebraten definiert, wie in Abb. 1.2 zu sehen.

Der Planer findet je nach den Voraussetzungen in der Küche dann eine Folge von Aktiväten, um das durch eine Konjunktion von Fakten beschriebene Teilziel des gesamten Rezepts umzusetzen (siehe Abb. 1.3). Das Beispiel illustriert, wie der Planer anhand seiner Kenntnis über die Küche, in der das Gericht zubereitet wird, alle detaillierten Schritte plant, die für das Teilziel `BRAETER-VORBEREITEN` notwendig sind, ohne dass sie jedoch im Rezept festgehalten wären.

```
(:action braeter-vorbereiten
 :parameters (?schritt)
 :precondition
 (and (schritt ?schritt)
      (= ?schritt braeter-vorbereiten)
      (in-bearbeitung ?schritt)
      (neuer-inhalt lorbeerblaetter braeter ?schritt)
      (neuer-inhalt pimentkoerner braeter ?schritt)
      (in-scheiben zwiebel zwiebelwuerfel ?schritt)
      (neuer-inhalt-sit-arg zwiebelwuerfel braeter ?schritt)
      (neuer-inhalt-sit-arg fleisch braeter ?schritt)
 )
 :effect (ausgefuehrt ?schritt)
)
```

Abb. 1.2: Planoperator in einem Kochrezept

Ähnlich wird Wissen über die Domäne für ROBERT repräsentiert, allerdings als HTN-Planoperatoren, d.h. als kleine Bäume, die Teiltasks des Gesamtproblems lösen (siehe dazu Kapitel **??** und [22, 3.5.2]). In beiden Beispielen gibt es ontologische Herausforderungen: eine systematische Taxonomie an Begriffen, die auch logisch konsistent sein muss, ist erforderlich, um Ad-Hoc-Benennungen zu vermeiden, die weder aus der Perspektive des Software-Engineering für Planungsprobleme noch aus der Perspektive des ontologischen Schließens adäquat sind.

Leider kann mit diesem Konzept nicht für jeden Task eine Problemlösung ermittelt werden. Denn Planungssprachen sind ja nicht turing-äquivalent, sondern wesentlich weniger ausdrucksstark. Sie basieren auch auf der *closed world assumption*, d.h. dass zu Planungsbeginn sämtliche relevante Information bekannt ist – in interaktiven Szenarien kann das aber nicht immer gewährleistet werden. Unsichere Information bedeutet aber ein exponentielles Wachstum der möglichen Zielzustände, was jedes System an seine Leistungsgrenzen bringt. Denn ist ein Fakt unsicher, kann er wahr oder falsch sein. Dies bedeutet, dass jeder unsichere Fakt zwei potentiell mögliche Zielzustände nach sich zieht. Wenn nun mehrere Fakten unsicher sind, wächst die Zahl der möglichen Zielzustände exponentiell mit der der unsicheren Fakten über die aktuelle Situation.

Viele Anwendungsprobleme sind auch gar keine Planungs-, sondern vielleicht Constraint-Satisfaction, Optimierungs- oder Klassifikationsprobleme, so dass ein Planer gar nicht das geeignete Instrument zur Berechnung einer Problemlösung ist, stattdessen aber Recommender-, Decision Support, Constraint-Solver-, Optimierungs- oder Klassifikationssysteme. Für unser Konzept ist das nicht schädlich, solange wir Zustände und Ergebnisse der benutzten Systeme symbolisch beschreiben können.

```
SCHRITT-BEGINNEN BRAETER-VORBEREITEN
HINZUFUEGEN FLEISCH BRAETER BRAETER-VORBEREITEN
AUS-SCHRANK-HOLEN ZWIEBEL BRAETER-VORBEREITEN
BEWEGEN ZWIEBEL HAND BRETT BRAETER-VORBEREITEN
ZWIEBEL-SCHNEIDEN ZWIEBEL ZWIEBELWUERFEL BRAETER-VORBEREITEN
AUS-SCHRANK-HOLEN LORBEERBLAETTER BRAETER-VORBEREITEN
HINZUFUEGEN LORBEERBLAETTER BRAETER BRAETER-VORBEREITEN
ABLEGEN LORBEERBLAETTER ARBEITSPLATTE BRAETER-VORBEREITEN
AUS-SCHRANK-HOLEN PIMENTKOERNER BRAETER-VORBEREITEN
HINZUFUEGEN PIMENTKOERNER BRAETER BRAETER-VORBEREITEN
BEWEGEN PIMENTKOERNER HAND SCHRANK BRAETER-VORBEREITEN
BEWEGEN ZWIEBELWUERFEL BRETT HAND BRAETER-VORBEREITEN
HINZUFUEGEN ZWIEBELWUERFEL BRAETER BRAETER-VORBEREITEN
BRAETER-VORBEREITEN BRAETER-VORBEREITEN
SCHRITT-BEENDEN BRAETER-VORBEREITEN
```

Abb. 1.3: Lösung für das Teilziel aus Abb. 1.2

1.4.4 Wissen über Nutzer, Interaktion und Kooperation

Viel wichtiger ist, dass für eine zielführende Lösung der oben genannten Probleme zusätzliches Wissen erforderlich ist [9]: über Kompetenzen in der Domäne sowie im Umgang mit einem Assistenzsystem, über Präferenzen, die Entscheidungen über mögliche Handlungs- oder Auswahl-Optionen beeinflussen, über den emotionalen Status, über Sprachverwendung und Interaktionsstil, um mit jedem Nutzer optimal kooperieren zu können.

Dieses Wissen ist oft zur Ermittlung einer Problemlösung hilfreich: sie soll nicht (nur) in der aktuellen Situation logisch korrekt, sondern auch optimal im Bezug auf Präferenzen des Nutzers sein. D.h. sie soll möglichst diejenigen Entscheidungen treffen, die ein Nutzer auch selbst treffen würde: ein Reiseassistenten, der einen Stadtbummel durch Rom plant (davon gibt es sehr viele), soll möglichst viele antike Stätten in die Tour einbauen, wenn der Nutzer eine Präferenz für Geschichte hat, möglichst viele enge Gassen und malerische Innenhöfe, wenn er bevorzugt unwechselbare Orte sehen möchte, möglichst viele barocke Kirchen, wenn sich der Nutzer für diese Architekturepoche interessiert, oder abends viele kleine Lokale mit regionaler Küche in Gegenden mit wenig Touristen, wenn der Nutzer Italienisch spricht und erleben möchte, wie die Römer versuchen, einen Abend im Restaurant unter sich zu verbringen. Präferenzen können aber je nach Stimmung variieren; deswegen kann es während der Problemlösung wichtig sein, die Stimmung online erfassen zu können [57]. Je nach Anwendung gibt es viele weitere Randbedingungen, die in eine Problemlösung eingehen können. Zu bemerken ist hierbei aber, dass die grundsätzliche Struktur der Problemlösung für den gegebenen Task schon bekannt ist (wie es auch in Abb. 1.2) der Fall ist.

Natürlich ist das Wissen über Tasks und Aktivitäten, mit dessen Hilfe Problemlösungen gefunden werden, schon ein wesentlicher Teil des Wissens über die Kooperation zwischen Nutzer und Assistenzsystem. Wie aber wie weiter oben bereits besprochen, benötigt ein Assistenzsystem auch Wissen darüber, wie es reagieren kann, wenn bei der Durchführung einer Aktivität nicht die von der Problemlösung vorgesehene Situation erreicht wird. Einerseits kann aus dem Wissen, in welcher Phase sich eine Aktivität gerade befindet, nach der Taxonomie in Tab. 1.1 der Unterstützungsbedarf klassifiziert werden. Andererseits kann, wenn die erreichte Situation identifizierbar ist, aus der Diskrepanz zwischen erwarteter und tatsächlicher Situation der *Fehler* in Bezug auf die Problemlösung bestimmt werden – also das *Symptom*. Mit entsprechenden Diagnoseregeln [54] können verschiedene Ursachen erschlossen werden. Für jede von ihnen kann es eine passende Assistenzstrategie geben, die entweder durch einen *Reparaturplan* [6] den ursprünglich erwarteten Zustand herstellen oder eine neue Problemlösung für das aktuelle gemeinsame Ziel berechnen kann, sofern es eine solche überhaupt (noch) gibt. Falls aber der aktuelle Task unlösbar geworden ist, muss das Assistenzsystem seine Problemlösung für den Kooperationstask anpassen durch eine geeignete Assistenzstrategie, die darin bestehen kann, wieder in die Phase der Motiv- und Zielbildung (siehe Tab. 1.1) einzutreten und einen Zielwechsel anzuregen. Eine andere Strategie, die weniger spektakulär ist, besteht im Abbruch der Kooperation – wie dies fast alle Programme machen, wenn sie eine Fehlermeldung ausgeben und darauf warten, dass der Nutzer eine neue Aufgabe formuliert.

Auch die Interaktion mit dem Nutzer ist ein Task für sich – hierbei bestehen Problemlösungen in der Planung einer Folge von Sprechhandlungen. Auch sie haben erwartete Effekte, die manchmal nicht eintreten. Beispiele dazu haben wir bereits oben beim Dialog mit dem Küchenhelfer gesehen. Assistenzstrategien beim Sprachverstehen sind also Tasks für Unterstützungsbedarf in diesem Bereich. Sie können analog zu Tasks für Kooperation und Domäne formalisiert werden (siehe die Erörtung oben zur Modellierung des Sprachverstehens von ROBERT).

Ein Assistenzsystem muss also immer aktuelle Problemlösungen für alle Aspekte der interaktiven und kooperativen Assistenz gleichzeitig verfolgen und deren Fortschritt protokollieren, um auf die aktuelle Situation und den Inhalt neuer Äußerungen des Nutzers sachlich plausibel und für den Nutzer nachvollziehbar reagieren zu können. Die Dialogbeispiele mit dem Küchenhelfer belegen, dass Nutzer ein derartiges Verhalten erwarten, weil sie es von der Kooperation mit anderen Personen gewöhnt sind.

1.5 Assistenz per Design

Ein Assistenzsystem entsteht aber nicht nur alleine dadurch, dass zu einem bestehenden technischen System eine Komponente – im Sinn der Companion-Technologie – hinzugefügt wird, die Reasoning über die Kompetenzen des technischen Systems

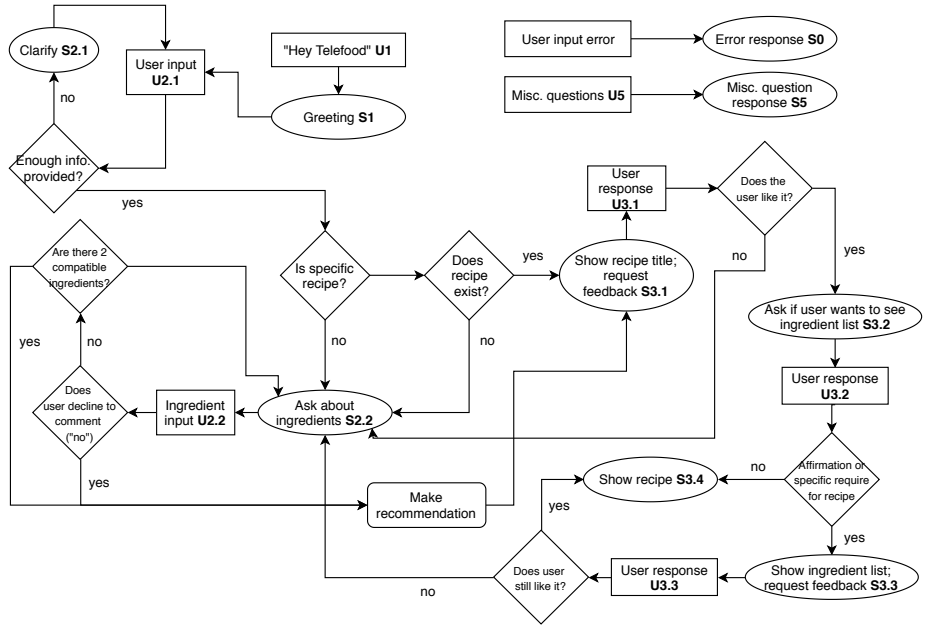

Abb. 1.4: Prozessmodell für Kooperation der der Suche nach einem passenden Rezept

betreiben kann, so wie oben ausgeführt. Vielmehr muss mit den gängigen Methoden des *requirement* und *software engineerings* analysiert werden, wie die Kooperation zwischen Nutzern und einem technischen System am besten gestaltet wird. Dabei bildet sich ein Verständnis dafür, welche Funktionalitäten das technische System als Aktivitäten im Sinn von oben zur Verfügung stellen muss (darüber wird kein Reasoning betrieben, weil sie die Basiskompetenzen bilden, mit denen der Agent zur Lösung von Aufgaben der Nutzer beitragen kann). Weiterhin wird ermittelt, welche Aktivitäten Nutzer durchführen, und wie und worüber Informationen zwischen Nutzern und Agent ausgetauscht werden.

Das Ergebnis einer derartigen Analyse kann ein Prozessmodell dafür sein, wie Problemlösungen für die geplante Anwendungsdomäne zu gestalten sind. Ein Beispiel dafür ist das Flussdiagramm in Abb. 1.4, in dem veranschaulicht ist, wie Nutzer mit dem Küchenhelfer kooperieren, wenn er sie bei der Auswahl von Rezepten unterstützen soll.

1.5.1 Strukturierung von Software hinsichtlich ihrer Funktionalität – Ausführungsmodelle

Es geht uns hierbei gar nicht um Details des Modells, sondern darum, deutlich zu machen, dass sich der Kooperationsablauf wesentlich davon unterscheidet, welche

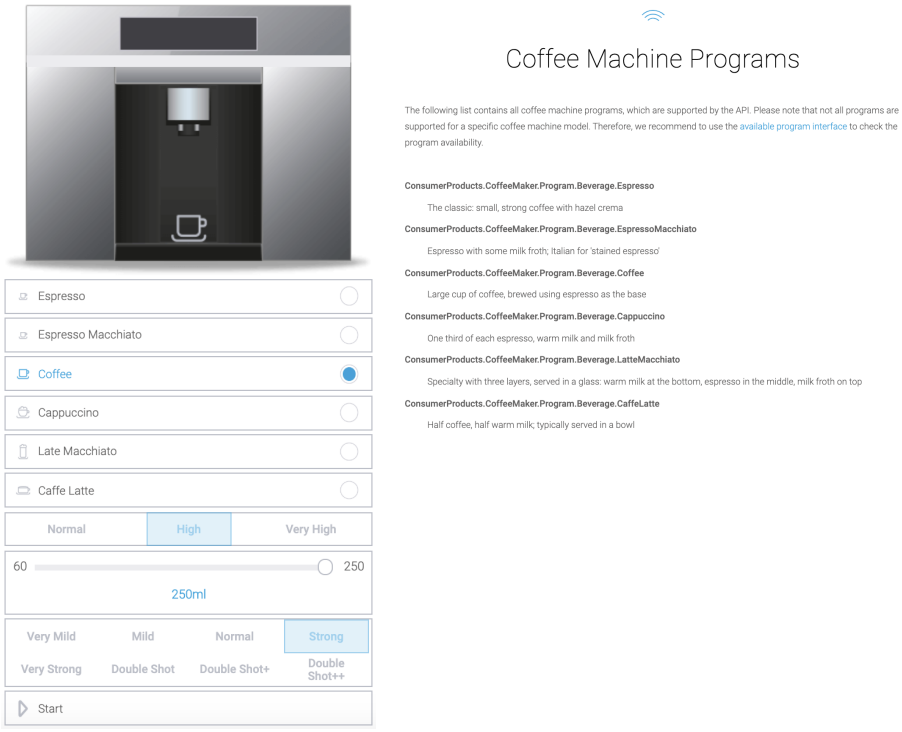

Abb. 1.5: API eines Kaffeevollautomaten

elementaren Funktionen das technische System bereitstellt, und wie diese in einfacher Weise von Nutzern für die Lösung ihrer eigentlichen Aufgabe verwendet werden können. Wir möchten stattdessen Problemlösungen für komplexe Aufgaben von Nutzern und sprachgesteuerte Interfaces für technische Systeme, bei denen Problemlösungen identisch mit der Auslösung bestimmter Funktionalitäten sind, gegenüberstellen. Ein Beispiel dafür ist der Simulator eines Kaffeevollautomaten, der auf der Webseite https://developer.home-connect.com/simulator/coffee-machine bereitgestellt wird[10].

Einfache Aufgaben sind, wie am Befehlsumfang von Abb. 1.5 ersichtlich, durch eine einzige Aktivität des technischen Systems lösbar: z.B. kann die Aufgabe, eine Tasse Kaffee zuzubereiten, durch die Aktivität `ConsumerProducts.CoffeeMaker.Program.Beverage.Coffee` gelöst werden. Das dargestellte Interface kann also als Mittel zur Kommunikation von Zielzuständen (nämlich dem Effekt von Aktivitäten verstanden werden). Komplexere Aufgaben wie das Zubereiten eines Espresso mit normaler Temperatur, eines Espresso mit wenig starkem Kaffee, zwei normalen Cappuccini und einem großen Latte Macchiato bedürfen der Planung einer geeig-

10 letzter Abruf: 20.10.2019

neten Lösung: Es sind nicht nur die Werte für die Parameter Temperatur, Menge, Stärke jeweils geeignet zu wählen, sondern je nach vorhandener Menge an Bohnen, Wasser und Milch sowie je nach Füllstand des Tresterbehälters muss die Problemlösung Aktivitäten des Nutzers zum Nachfüllen bzw. Entleeren der Vorratsbehälter einplanen.

Die Zahl der Fakten, die einen Zielzustand für eine komplexe Aufgabe beschreiben, ist also sehr groß und sogar zu groß dafür, in einem graphischen User Interface verständlich dargestellt werden zu können. Ein User Interface für solche komplexen Problemlösungen wäre daher gar nicht sinnvoll zu gestalten, auch wenn sie in der Anwendungsdomäne häufig sind: kommt Besuch nach Hause, oder findet eine Besprechung im Büro statt, möchten oft mehrere Personen ganz unterschiedliche Varianten von Kaffee.

Nicht nur das User Interface wäre kompliziert, sondern auch die Implementierung der Steuerung für den Kaffeevollautomaten. Deshalb ist die Idee der Companion-Technologie sehr hilfreich: Bedienabläufe werden in Abstimmung mit der elementaren Funktionalität, aber auch erst nach sorgfältiger Analyse als separate Komponente mit eigenem Domänenwissen, das zu dem des technischen Systems komplementär ist, entwickelt.

1.5.2 Interaktionsmodelle für Assistenzsysteme

Ähnliches gilt für die HCI-Komponenten eines Assistenzsystems. Wie schon an den Dialogen mit dem Küchenhelfer aus Sicht der Nutzer, aber auch am Beispiel des Kaffeevollautomaten deutlich wurde, kann sich ein Interface, das die Kommunikation über komplexe Problemlösungen ermöglichen soll, nicht auf elementare Funktionalitäten beschränken, sondern muss alle Aktivitäten einer Problemlösung umfassen.

Idealerweise realisiert ein UI gerade die Kommunikations-Vorgänge, die für eine geplante Problemlösung erforderlich sind. Damit können Assistenzstrategien für die einzelnen Phasen aus der Taxonomie nach Wandke implementiert werden, wenn sie ausschließlich kommunikative Mittel einsetzen. Dies kann eine andere Interface-Gestaltung erforderlich machen als sie für ein an den Funktionalitäten des technischen Systems orientiertes UI erforderlich ist. Es sind nicht mehr die Parameter des technischen Systems wie in Abb. 1.5 festzulegen, sondern die der Aktivitäten aus der Problemlösung. Aus ihnen können dann die Funktionalitäten parametrisiert werden.

Ein Beispiel dafür ist in Abb. 1.6 zu sehen. Sie visualisiert Schritte aus einer Problemlösung für die Herstellung eines Gerichts. Jede Zeile des Display informiert über einen Schritt in der Problemlösung. Das technische System besteht dabei nur in einzelnen Elektrogeräten (wie in Abb. 1.5), das Assistenzsystem muss den Nutzer aber auch bei vielen anderen Aktivitäten unterstützen können (z.B. bei der Auswahl eines geeigneten Topfs). Insbesondere muss es, wie oben schon besprochen, darauf reagieren

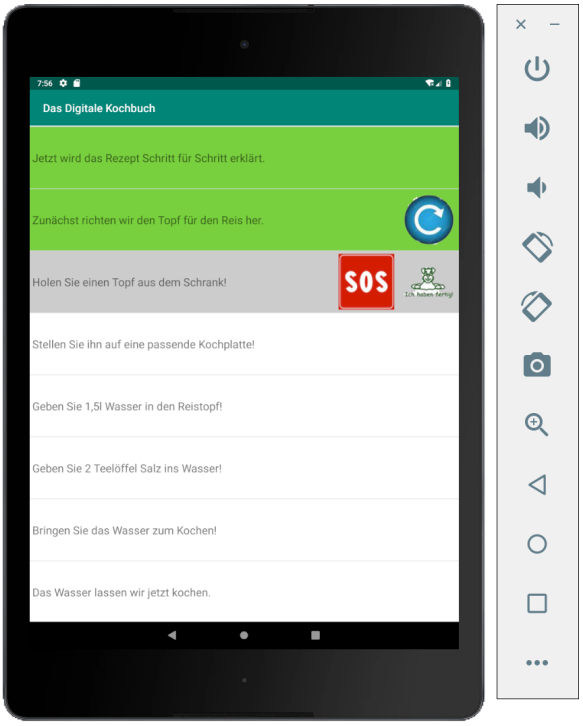

Abb. 1.6: Screenshot eines Küchenhelfers, der bei der Herstellung eines Gerichts assistiert

können, dass der Nutzer mit der Durchführung einer Aktivität nicht zurecht kommt und dann beispielsweise visuelle Erklärungen anbieten, die mit den Komponenten des technischen Systems gar nichts zu tun haben.

Ein Assistenzsystem muss sich oft proaktiv verhalten können. Im Szenario der Unterstützung beim Kochen ist dies z.B. dann erforderlich, wenn das System annehmen kann, dass Nutzer auf ein wichtiges Ereignis (z.B. Nudeln sind fertig gekocht) nicht reagieren. Um das Scheitern einer Aktivität zu verhindern, versucht das System dann durch Strategien der Motiv- und Zielbildung den Nutzer zur Durchführung der notwendigen Aktivitäten zu motivieren.

Weitere Motivationen für proaktives Verhalten ergaben sich aus Experimenten mit dem Assistenzsystem ROBERT [5]. Bei der Evaluation wurde deutlich, dass den Nutzern oft nicht klar war, auf welche Weise sie mit dem System interagieren können. Dies lag vor allem an der Vielzahl von Möglichkeiten, mit der Anwendung zu interagieren. Diese konnte vier verschiedene Themenbereiche für Interaktion abdecken: Navigation, Planungsanfragen, fachliche Erklärungen und Assistenzstrategien (Video, Überblick). Bei rein passivem Verhalten war es dem System nicht möglich, dem Nutzer seine Funktionalität vollständig zu offenbaren. Um Nutzer das Potenzial der

Kooperation erfahrbar zu machen, ist es also in komplexen Systemen, wie beispielsweise ROBERT oder dem beschriebenen Küchenassistenten, die aktiv Erklärungen über die eigene Funktionsweise anzubieten, um den Dialog mit dem Nutzer zu vereinfachen und ihm den Einstieg in den Assistenten zu vereinfachen.

Ein weiteres Problem lag in der beschränkten Kontexterkennung der derzeitigen Assistenzsysteme. So war beispielsweise ROBERT nur in der Lage, die direkte Interaktion (Nutzeräußerungen, Navigation des Assistenten) mit dem Nutzer nachzuverfolgen. Die Variante von *state tracking* stößt aber schnell an Grenzen, da das System, wenn Nutzer von der Beschreibung einer Teilaufgabe zu der einer anderen wechselte, davon ausging, dass die Teilaufgabe auch erfolgreich abgeschlossen werden konnte. So wird beispielsweise nicht erkannt, ob ein Nutzer sich nur einen Überblick über die nachfolgenden Aufgaben machen will oder mit diesen beginnen möchte. Zudem werden die Probleme mit der Aufgabe an sich nicht automatisch erkannt. Ein Nutzer kann nur von sich aus Probleme schildern, und der Assistent versucht zu helfen. Dabei ist er oder sie aber möglicherweise gar nicht bewusst, was genau das Problem ist oder wie das Problem systemverständlich auszudrücken ist. Um derartiges „Alleine gelassen werden“ zu vermeiden, ist proaktives Nachfragen seitens des Assistenzsystems notwendig um dem Nutzer die Problematik mit der Aufgabe zu erleichtern und nicht durch passives Verhalten zu einer weiteren Hürde zu werden.

Zur besseren Verständlichkeit des Konzepts Proaktivität in der Mensch-Maschine-Interaktion sei hier eine kurze Einführung gegeben. Der Begriff Proaktivität ist weit verbreitetet im Bereich der Arbeits- und Organisationspsychologie [23]. Gemäß der Definition geht es bei proaktivem Verhalten darum, die Kontrolle zu übernehmen, problematische Situationen zu antizipieren und zu verhindern, anstatt nur passiv zu bleiben und zu beobachten. [45] beschrieben die Proaktivität in Dialogsystemen als ”ein autonomes, vorausschauendes, systeminitiiertes Verhalten, mit dem Ziel, vor der zukünftigen Situation zu agieren, anstatt nur darauf zu reagieren“. Es ist nachgewiesen, dass Proaktivität in der Mensch-Mensch-Interaktion die Sozialisierung und die Leistung am Arbeitsplatz beeinflusst [11]. Aktuelle Untersuchungen zu proaktivem Verhalten in der Mensch-Maschine-Interaktion deuten auch darauf hin, dass aktive Systemaktionen einen Einfluss auf die Wahrnehmung des Systems durch den Nutzer haben [47].

Die Modellierung von akzeptierten und vertrauenswürdigen proaktiven Strategien ist jedoch noch eine offene Frage. [45] definierten drei Herausforderungen für proaktive Dialogsysteme: ob ein System überhaupt proaktiv sein muss, sowie wie und wann es die Initiative ergreifen sollte. Die meisten aktuellen Forschungsarbeiten konzentrieren sich darauf, wie man proaktives Verhalten basierend auf dem Autonomiegrad von Systemen gestaltet [47]. Der richtige Zeitpunkt, um geeignete Strategien auszulösen, wird jedoch stark unterbewertet. Wann ein System proaktiv sein sollte, ist ein heikles Thema. Das Einleiten einer Interaktion zu einem ungeeigneten Zeitpunkt könnte als störend und aufdringlich empfunden werden. Dies kann die Mensch-Computer-Beziehung

beeinträchtigen, insbesondere im Hinblick auf das wahrgenommene Vertrauen des Benutzers in das System [29].

Weitere Herausforderungen für intuitive Interaktion fanden sich bei ROBERT beim Sprachverständnis und im Dialogmanagement. Zur Aktivierung des Sprachassistenten musste ein Nutzer jede seiner Äußerungen mit dem Schlüsselwort "Robert" beginnen. Ziel dieses Ansatzes war es zu vermeiden, dass ROBERT auf nicht assistentenbezogene Äußerungen, so genanntem "Off-Topic-Talk" reagiert, da die Spracherkennung standardmäßig dauerhaft aktiviert war. Für die Dialogführung hatte dies jedoch den Nachteil, dass der Nutzer während jedem Dialogschritt das Schlüsselwort benutzen musste, was eine unnatürliche Interaktion zur Folge hatte. Zur Vermeidung dieses Problems ist eine automatische Erkennung notwendig, wann ein Nutzer seine Äußerungen an das System richtet und wann nicht. Forschung dazu findet im Bereich der Spracherkennung zum Thema *off topic detection* [43] statt. Eine andere Möglichkeit zu erkennen, wann Nutzer mit einem System interagieren wollen oder wann sie sinnvollerweise vom System angesprochen werden, ist die Blickanalyse [10]. Menschen beginnen eine Kommunikation untereinander oft mit der Aufnahme von Blickkontakt. Durch Detektion der Blickrichtung mit dem Ziel, herauszufinden, ob ein Nutzer den Agenten — auf einem Display visualisiert, können geeignete Zeitpunkte zur Identifikation ermittelt werden [50, 55, 48]. Allerdings ist Blickanalyse ohne Hardware wie Eyetracker-Brillen, die Nutzer tragen müssen, schwierig. Eyetracker am Display müssten immer geeignet kalibriert werden. Erfolgversprechender sind Softwarelösungen, die Video-Streams analysieren, um Gesichter und dann darin die Pupillen zu finden. Daraus kann dann eine Blickrichtung geschätzt werden. Eine freie Softwarelösung für diese Aufgabe ist OpenFace [3].

Eine weitere Problematik bei der Entwicklung von Assistenzsystemen stellt die Komplexität der Dialogmodellierung dar. Durch die Interaktion in verschiedenen Themenbereichen und weil das System Nutzer durch eine Vielzahl von Teilschritten einer Aufgabe führt, entsteht eine hohe Anzahl an möglichen Dialogpfaden, wenn man aktuelle Tools wie RASA oder snips.ai zur Dialogmodellierung verwendet. Dies stellt eine enorme Herausforderung für den Dialogdesigner dar, da sämtliche Möglichkeiten berücksichtigt werden müssen, um zu entscheiden, wie das System wann reagieren soll. Dieses Problem beschränkt sich aber nicht nur auf das Dialogmanagement, sondern betrifft auch die Modellierung der Sprachverständniskomponente. Durch die möglichen Interaktionsthemen ist eine hohe Varianz von möglichen Nutzeräußerungen zu erwarten, welche heutige Sprachkomponenten kaum verarbeiten können. Um den Sprachbereich des Nutzers einzugrenzen, ist eine Einteilung der Modelle nach verschiedenen Themenbereichen, beispielsweise bei ROBERT in Navigation und Interaktion, erforderlich. Außerdem kann mithilfe von „Priming", also durch wiederholtes Verwenden von Formulierungen in Systemäußerungen, auf die das System zu erwartende Antworten auch analysieren kann [59], der Nutzer implizit dazu bewegt werden, für das System verständliche Sprache zu verwenden.

Antworten darauf, wie der Komplexität der Dialogmodellierung durch die Identifikation einzelner Teilaufgaben und den Einsatz von geeigneten KI-Algorithmen begegnet werden kann, sollen im Folgenden besprochen werden.

1.6 Kooperationsmodelle für Assistenzsysteme

Die Diskussion bisher hat ergeben, dass ein Algorithmus zur Kontrolle eines Assistenzvorgangs den Status mehrerer Tasks gleichzeitig tracken muss:

- die Kooperation selbst, und dabei vor allem die Frage, ob die jeweils vom technischen System oder dem Nutzer ausgeführten einzelnen Aktivitäten den erwarteten Effekt erreicht haben;
- die Interaktion, falls eine aktuelle Aktivität ein Interaktionsvorgang war, und hierbei vor allem, ob der Informationstransfer (z.B. Frage beantwortet, Information bestätigt) erwartungsgemäß funktioniert hat
- die Aktivitäten der Problemlösung, und hierbei vor allem, ob die einzelnen Handlungsphasen erwartungsgemäß abgeschlossen werden konnten.

Nicht erfüllte Erwartungen können weitere Tasks auslösen, die der Planreparatur dienen bei Beibehaltung des ursprünglichen gemeinsamen Ziels oder der Zielrevision als mögliche Aktivität des Kooperationstasks, wenn keine alternative Problemlösung mehr gefunden werden kann. Ähnliche Kontrollschleifen werden in der Robotik genutzt [63]. Dabei spielt *belief tracking* eine wichtige Rolle [71, 35], und dies gilt auch für Assistenzsysteme: jede beobachtete Aktivität (sei es ein physikalischer oder ein Informationsvorgang) führt dazu, dass der Kontrollalgorithmus untersuchen muss, welchen der bekannten Tasks diese Aktivität fortführen möchte. Je nach Aktivität kann dies unterschiedlich schwierig sein: während die Betätigung eines Buttons in einer UI per Konstruktion eine eindeutige Semantik hat, ist dies bei natürlichsprachlichen Äußerungen nicht immer der Fall, wie die Beispiele oben gezeigt haben. Die Beobachtung eines physikalischen Vorgangs hingegen kann wiederum eindeutig sein, wenn das Signal dafür eindeutig ist (z.B. der Event der Motorsteuerung des Thermomix, dass er jetzt auf die angegebene Geschwindigkeitsstufe eingestellt ist, oder ob ein Recommender-System mindestens ein Rezept vorschlagen kann), schwierig aber, wenn das Signal mehrdeutig ist (z.B. ob ein Gesichtserkenner den Nutzer in der Küche hinreichend sicher identifizieren kann). Ähnliche Problem hat ein Roboter bei der Zuordnung von Sensorsignalen zu Objekten seiner (mentalen) Karte. Während aber in der Robotik für diese *grounding*-Probleme allgemeine und applikationsunabhängige Lösungen bekannt sind (z.B. SLAM), stellt das *belief tracking* für Assistenzsysteme im hier beschriebenen allgemeinen Sinn eine offene Forschungsfrage dar.

Stand der Technik ist hier, einen Dialogzustand, der dem Ausführungszustand der Problemlösung (repräsentiert als konstanter Zustandsübergangsgraph) entspricht, zu verfolgen. [42] konstruieren tiefe neuronale Netze, um einen Assistenztask, in dem eine

Reihe von Parameterwerten für eine Suchanfrage gesammelt wird, zu tracken. Mit dem Ansatz sollen verschiedene Formulierungen von Äußerungen in einem Assistenzdialog genau einem Knoten (d.h. Ausführungszustand) im Zustandsübergangsgraphen zugeordnet werden. Viele der Phänomene aus Dialogen zu Assistenztasks, die oben in den Beispielen besprochen wurden, werden so in der internen Repräsentation des neuronalen Netzes berücksichtigt. Eine explizite Repräsentation verschiedener Taskzustände, wie wir sie oben eingeführt haben, ist damit aber nicht möglich.

Einen großen Schritt in diese Richtung gehen [34] mit dem Konzept der *task lineages*: Es erlaubt, mehrere Tasks und deren Zustand parallel zu verfolgen und jede neue Äußerung zum Update eines der Tasks zu verwenden. Das Konzept wurde entwickelt, um das sogenannte *dialogue state tracking* zu implementieren, bei dem Dialoge bereits vorliegen, und der Algorithmus im Nachhinein den Ausführungszustand verfolgen und seine Historie über den ganzen Dialog rekonstruieren soll. Für das Führen neuer Dialoge mit einem expliziten, auch durch Handlungen der Kooperationspartner und Beobachtungen der Umgebung beeinflussten Kontext muss das Konzept gerade um die Fähigkeit, alle aktuellen nicht-linguistischen Kontextinformationen berücksichtigen zu können, erweitert werden. Wir werden diesen Punkt in Kapitel 1.7 weiter ausführen.

Bei verfügbaren, generischen Toolboxen zur Entwicklung von Chatbots, oder allgemeiner *conversational agents*, wird kein expliizites *reasoning* über Tasks betrieben, sondern sie werden mit der Konfiguration festgelegt. Beispielsweise wird mit der Toolbox RASA[11] eine Problemlösung für einen Agenten in einer sog. *story* festgelegt, die, ähnlich wie ein einfacher endlicher Automat zulässige Folgen von Zustandsübergängen beschreibt. Für die Kaffeemaschine in Abb. 1.5 definiert folgende *story*, wie Nutzer durch Interaktion einen Kaffee bekommen können:

```
## request coffee
* greet
  - utter_greet
* make_espresso
  - utter_feedback_espresso
  - home_connect_coffee_maker_program_beverage_espresso
  - utter_did_that_help
```

Mit Sternchen beginnen sog. `intents`, das sind Bezeichner für Aktivitäten, über die allerdings kein Domänenwissen modelliert ist, sondern für die nur alternative Formulierungen vorliegen, z.B. für `make_espresso`:

```
## intent:make_espresso
 - I would like an [espresso](beverage_type), please.
```

11 https://www.rasa.com

```
- [espresso](beverage_type)!
- Could you make an [espresso](beverage_type) for me?
- make me an [espresso](beverage_type)!
```

Die NLU-Komponente von RASA und ähnlichen Tools implementiert Algorithmen zur *entity extraction*, die erstens aus einer Äußerung *entities* wie `beverage_type` instantiieren und zweitens die Wahrscheinlichkeit berechnen, mit der `make_espresso` vom Nutzer gemeint sein kann. Mit dieser Wahrscheinlichkeit kann dann geschätzt werden, wie wahrscheinlich es ist, dass die *story* `request coffee` fortgesetzt wird. Dieses Verfahren kann als Beschränkung des Konzepts der *task lineages* auf einen Task interpretiert werden. Damit sind auch schon die Beschränkungen des Modells hinsichtlich seiner Fähigkeit, komplexe Dialoge zu verarbeiten, abgesteckt.

Im Folgenden werden wir daher ein Modell entwickeln, das diese Beschränkungen überwinden kann.

https://arxiv.org/pdf/1902.00771.pdf

1.7 Assistenz durch KI-Algorithmen

Algorithmische Modelle für die Verarbeitung natürlichsprachliche Dialoge haben eine lange Tradition in der Forschung. Einen guten Überblick darüber gibt [68]. Eine zentrale Linie hat einen linguistischen Fokus und nutzt Folgen von Sprechakten als theoretische Grundlage [26]. Ein anderer, an der Theorie rationaler Agenten orientierter Ansatz operationalisiert kommunikative Zwecke (Intentionen) [2, 25] und soziale Konventionen zur Interaktion [64]. Eine dritte Linie geht von Plänen und Zielen in der Domäne und für die Kommunikation als zentrales Konzept eines Dialogmodells aus [4].

Daraus und aus den vorausgegangenen Analysen wollen wir nun ein Konzept entwickeln, das die eben

Die Bei Forsspielchungen zusaselini für Interaerwähntenktionen mit dem Küchenhelfer in Abschnitt 1.3.3 belegen, dass in der Tat immer mehrere *task lineages* gleichzeitig aktiv sind, die auch eine graphartige Zusammenhangsstruktur haben. Der Zusammenhang zwischen einem Task t_1 und t_2 wird dabei vom Zweck der Interaktion (siehe Tab. 1.1 und die annotierten Dialogbeispiele) bestimmt. Er erklärt, warum eine neue Aktivität durchgeführt wird. Neben dem Zweck der Interaktion gibt es auch den Zweck der Kooperation, der letztlich die Interaktion motiviert (vergleiche die Liste der Problemlöse-Kompetenzen in Abschnitt 1.3.3):

- Ermitteln einer Problemlösung für einen Task
- Bestimmen des Fortschritts in jedem Task
- Ausführen einzelner Aktivitäten
- Diagnose von Assistenzbedarf
- Ausführen einer Assistenzstrategie

- Fortführen der aktuellen Problemlösung
- Abbrechen einer Problemlösung

Das Assistenzsystem muss nach jeder eigenen Aktivität und jeder beobachteten Aktivität des Nutzers entscheiden, welche Kompetenz aktuell die zweckmäßigste für die Unterstützung des Nutzers ist, und die dafür notwendigen Aktivitäten durchführen. Daraus ergibt sich dann der Zusammenhang zwischen Tasks; der entstehende Graph verallgemeinert das Konzept der *task lineages*.

Algorithmisch am einfachsten ist natürlich das *Fortführen der aktuellen Problemlösung*, weil dafür nur die nächste Aktivität im Plan für das gemeinsame Ziel ausgeführt werden muss. In anderen Fällen sind jeweils geeignete KI-Verfahren erforderlich, z.B. Planen und vor allem Verfahren zum *state tracking*: welche Diagnose trifft zu, welche Assistenzstrategie ist die beste, welchen aktuellen Status hat der Task — hierfür können symbolische Regelsysteme oder probabilistische Ansätze zum Schätzen eines Zustands geeignet sein.

Neben der Entscheidung über eigenes Handeln muss das Assistenzsystem aber auch das Verhalten des Nutzers entsprechend klassifizieren können: welchen Kooperationszweck verfolgt er, wie verdeutlicht er ihn in Interaktionen mit dem Assistenzsystem? Schon die Dialogbeispiele oben zeigen, dass diese Aufgabe sehr kompliziert ist, weil sowohl der Zweck der Kooperation als auch der Assistenzbedarf und Interaktionszweck nicht oder nur implizit gegeben werden. Wenn z.B. der Küchenhelfer im ersten Dialogbeispiel in A_4 die gefundenen Rezepte nennt, wird nicht explizit kommuniziert, dass es hier um das *Fortführen der Problemlösung* geht. Wenn der Nutzer zuvor in TP_3 äußert *Hab ja kein Fleisch*, dann beschreibt der explizit seine Perspektive über die aktuelle Situation und implizit dignostiert er Assistenzbedarf. Wenn er in TP_5 sagt, *Ok dann mach ich es anders* gibt er implizit an, dass der den aktuellen Task abbrechen möchte, explizit gibt er an, dass er einen neuen Task starten möchte.

Damit die hier aufgeführten und im Dialogbeispiel annotierten Zuordnungen von Bedarf und Zweck algorithmisch ermittelt werden können, ist zunächst für jeden Task sein aktueller Zustand zu registrieren. Er besteht aus

- dem gemeinsamen Ziel
- der gefundenen Problemlösung
- der aktuellen Aktivität
- dem aktuellen Fortschritt (Zustand) der Problemlösung
- der Phase, in der sich die Aktivität gerade befindet
- einer Diagnose, ob der für die Phase erwartete Effekt eingetreten ist
- einer Diagnose, ob der aus der Problemlösung erwartete Effekt für die aktuelle Aktivität eingetreten ist,
- der Relation zu einem anderen Task für den Interaktionszweck
- der Relation zu einem anderen Task für den Kooperationszweck.

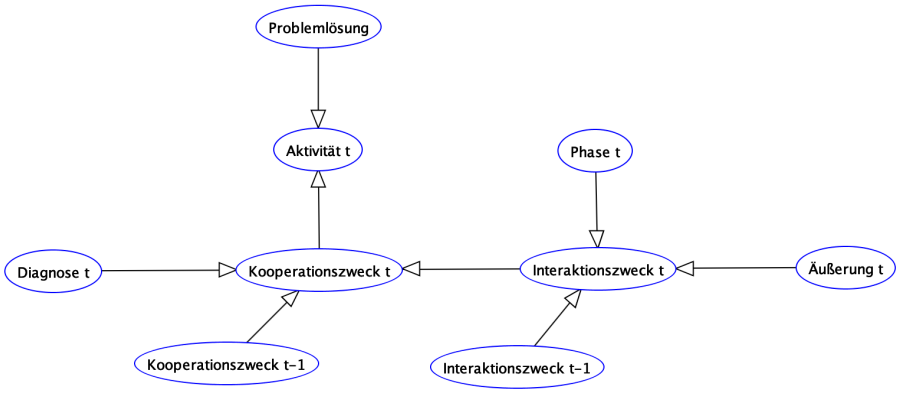

Abb. 1.7: Kausale Abhängigkeiten der einzelnen Aspekte des Zustands eines Tasks

1.7.1 Planung

Bis auf das gemeinsame Ziel, die Problemlösung dafür und die aktuelle Aktivität (bei der Planausführung) sind alle Aspekte des Task-Zustands mit Unsicherheit behaftet und können sich gegenseitig bedingen.

Für die Ermittlung von Problemlösungen können Planungsalgorithmen zum Einsatz kommen, wie wir in Abschnitt 1.4.3 ausgeführt haben.

Um das Problem des *state tracking* zu lösen, müssen für jeden Aspekt aktuelle Werte geschätzt werden. Diese Herausforderung ist vergleichbar mit dem SLAM-Problem in der Robotik [63, Abschnitt 10], bei dem gleichzeitig die aktuelle Position eines Roboters und seine Karte, die sein „Verständnis“ seiner Umgebung konstituiert, geschätzt werden.

In unserer Situation hingegen sind der aktuelle Kooperations- und Interaktionszweck (d.h. der zu befriedigende Assistenzbedarf nach Tab. 1.1) sowie die Phase der aktuellen Aktivität die latenten Variablen, die aus der aktuellen Äußerung und der aktuellen Diagnose über die Diskrepanz zwischen erwarteten und beobachteten Effekten zu schätzen sind. Damit können wir die Erläuterung oben konkretisieren: für TP_5 möchten wir als Kooperationszweck *Abbrechen einer Problemlösung* mit Relation zu TP_3, als Interaktionszweck *Grad der Zielerreichung bewerten* mit Relation zu A_4[12] schätzen. Als aktuelle Phase ergibt sich daraus *Effektkontrolle*.

12 und damit auch TP_3, da A_4 über den Kooperationszweck *Fortführen einer Problemlösung* und den Interaktionszweck *Information über ausgewählte Optionen* in Relation zu TP_3 steht

1.7.2 Erstellen von Diagnosen

Dabei ist das Erstellen der aktuellen Diagnose ein Problem für sich, das wir kurz in Abschnitt 1.8 ansprechen werden. Ausführlich diskutiert werden Verfahren zur symbolischen Inferenz bei [54], probabilistische Verfahren beruhen oft auf Bayes-Netzen [12] — beide können auch als Lern- und Prädiktionsproblem für tiefe Neuronale Netze interpretiert werden, wie beispielsweise in [28, 77] beschrieben. Ebenso stellt das Inferieren des Interaktionszwecks eine eigene Herausforderung dar (neben der Spracherkennung), auf die wir in Abschnitt 1.9 eingehen werden.

1.7.3 Probabilistische Inferenz des Nutzerstatus

Zunächst aber wollen wir herausstellen, wie die latenten Variablen mit Hilfe eines rekursiven Filters geschätzt werden können (analog eben zum Prinzip von SLAM). Dazu sei k_t der Kooperationszweck zum Zeitpunkt t, i_t der Interaktionszweck, $a_{1:t}$ die Historie aller Aktivitäten seit Beginn der Kooperation, $d_{1:t}$ alle bisher ermittelten Diagnosen und $m_{1:t}$ alle bisherigen Interaktionen (natürlichsprachlich oder in einer anderen Modalität). Zur Bestimmung von k_t und i_t suchen wir diejenigen Werte, die

$$P(k_t, i_t | a_{1:t}, d_{1:t}, m_{1:t})$$

maximieren. Zur Entwicklung eines rekursiven Filters dafür überlegen wir zunächst, welche Ursache-Wirkung-Beziehungen zwischen den beteiligten Variablen bestehen und stellen sie als graphisches Modell (siehe Abb. 1.7) dar. Daraus können wir Annahmen über bedingte Unabhängigkeiten ableiten, die uns die folgende rekursive Berechnung der zu optimierenden Größe ermöglichen:

$$\begin{aligned}
& P(k_{t+1}, i_{t+1} | a_{1:t+1}, d_{1:t+1}, m_{1:t+1}) \\
= \; & \frac{P(k_{t+1}, i_{t+1}, a_{1:t+1}, d_{1:t+1}, m_{1:t+1})}{P(a_{1:t+1}, d_{1:t+1}, m_{1:t+1})} \\
= \; & \frac{\sum_{k_t} \sum_{i_t} P(k_{t+1}, k_t, i_{t+1}, i_t, a_{1:t+1}, d_{1:t+1}, m_{1:t+1})}{P(a_{1:t+1}, d_{1:t+1}, m_{1:t+1})} \\
= \; & \sum_{k_t} \sum_{i_t} P(k_{t+1} | k_t, i_{t+1}, a_{t+1}, d_{t+1}) \cdot P(i_{t+1} | i_t, m_{t+1}) \cdot P(k_t, i_t | a_{1:t}, d_{1:t}, m_{1:t})
\end{aligned}$$

Dabei ist $P(k_{t+1} | k_t, i_{t+1}, a_{t+1}, d_{t+1})$ das Kooperationsmodell, das den Kooperationszweck k_{t+1} vorhersagt, ihn mit dem Interaktionszweck i_{t+1} und mit der bisherigen Schätzung für k_t, i_t gewichtet.

Nach der Ermittlung der wahrscheinlichsten Werte für k_t und i_t haben wir eine aus seiner Äußerung m_t und dem aktuellen Kontext abgeleitete Hypothese dafür, wie der Nutzer vom Fortschritt der Kooperation einschätzt (siehe oben). Die

Einschätzung des Agenten ermitteln wir, wenn wir statt der Äußerungen $m_{1:t}$ die Beobachtungen des Agenten $o_{1:t}$ durch seine Sensoren heranziehen. Interpretieren beide Kooperationspartner die aktuelle Situation identisch, wird der Agent versuchen, den identifizierten Assistenzbedarf durch Auswahl einer geeigneten Strategie zu beseitigen. Wenn die Situation unterschiedlich bewertet wird, wird das Assistenzsystem eine Strategie auswählen, die die unterschiedliche Bewertung auflösen kann. Hierfür ist wieder TP_3 ein gutes Beispiel: Mit A_2 kommuniziert der Agent implizit, dass der erwartete Effekt für die aktuelle Aktivität, nämlich eine Reihe von Vorschlägen für umsetzbare Rezepte, eingetreten ist, und reagiert auf den aus TP_1 identifizierten Assistenzbedarf durch verbale *Information über ausgewählte Optionen*. An der Äußerung TP_3 muss er jedoch erkennen, dass der Nutzer *negative Rückmeldung* gibt und damit auf der Kooperationsebene die *Diagnose von Assistenzbedarf* bezweckt, während der Agent selbst das *Fortführen der Problemlösung* erwartet hatte. Der Agent kann nun durch Integration der Information aus TP_3 den Zustand des Tasks aktualisieren (es ist jetzt bekannt, dass Fleisch fehlt) und durch die Strategie des *Abbrechens der Problemlösung* und des *Ermittelns einer Problemlösung für einen neuen Task* versuchen, die Kooperation aufrechtzuhalten. Daraus geht A_4 hervor.

1.7.4 Ermitteln optimaler Strategien

Bei der Diskussion der Dialogbeispiele bisher war oft davon die Rede, dass ein Agent eine Strategie wählen muss, die potenziell in der Lage ist, einen erkannten Assistenzbedarf zu befriedigen. Ein mathematisches Modell für dieses Problem ist die Darstellung des Auswahlvorgangs als *partially observable decision process* (POMDP). Die Grundlagen dafür sind in [51] dargestellt. Für uns ist daran interessant, dass bei einem POMDP davon ausgegangen wird, dass der Zustand des Prozesses oder Tasks, in dem eine Auswahl für die beste nächste Aktivität eines Agenten (sog. *policy*) zu treffen ist, nicht direkt, sondern nur anhand von typischen „Symptomen“ des Zustands beobachtbar ist. Derselbe Sachverhalt liegt bei der Schätzung des aktuellen Kooperations- und Interaktionszwecks durch den oben beschriebenen rekursiven Filter vor: k_t und i_t werden nicht direkt beobachtet, sondern anhand einer Äußerung des Nutzers und einer aus Beobachtungen der Umgebung durch Sensoren gewonnenen Diagnose. Anhand dieser unsicheren Information wird der eigentliche Zustand eines Tasks, nämlich der aktuelle Fortschritt der zugeordneten Problemlösung geschätzt. Abhängig von dieser Schätzung des Zustands sucht ein POMDP-Prozess diejenige *policy* aus, die den zu erwartenden Nutzen für die komplette Abarbeitung der ermittelten Problemlösung maximiert.

Dies funktioniert mit dem Zustandsmodell für einen Task, ob die aktuelle Aktivität erfolgreich vollständig ausgeführt und ob sie die letzte Aktivität der Problemlösung ist. Aus den Schätzungen für k_t und i_t kann dann eine Wahrscheinlichkeitsverteilung über die möglichen Zustände eines Tasks geschätzt werden (*belief update*, siehe [63,

Abschnitt 16]). Auf dieser Basis kann — wie zu Beginn von Abschnitt 1.7 erläutert — diejenige Kompetenz ausgewählt werden, die den zu erwartenden Nutzen im Planungshorizont maximiert. Dabei ist der Nutzen davon, eine Kompetenz anzuwenden, sehr hoch, wenn sie erreicht, dass die letzte Aktivität der Problemlösung erfolgreich vollständig ausgeführt werden kann. Der Nutzen andere Aktivitäten ausgeführt zu haben, ist kleiner; verschwindend ist der Nutzen, eine Assistenzstrategie auszuführen, weil sie den Kooperationsablauf verlängert.

Das Ausführen einer Assistenzstrategie bedeutet, einen neuen Task zu starten, dessen gemeinsames Ziel im Effekt der Aktivität liegt, bei deren Ausführung Assistenzbedarf entstanden ist. Sobald dieser Task erfolgreich ausgeführt ist, kann auch der zu ihm in Relation stehende ursprüngliche Task weiter bearbeitet werden.

1.7.5 Deep Learning mit explizitem Wissen als Lösung?

Sowohl der rekursive Filter als auch das POMDP-Modell für die Strategieauswahl enthalten Lernaufgaben, die mit Hilfe von Methoden des maschinellen Lernens gelöst werden können [75, 73, 69]. Neben klassischen Methoden (wie *expectation maximization*) besteht eine Möglichkeit, die notwendigen Wahrscheinlichkeitsverteilungen zu schätzen darin, geeignete (tiefe) neuronale Netze zu trainieren. Dies gilt auch für die Wert-Iteration, mit der die Bewertung von *policies* gelernt wird.

Ein anderer Ansatz besteht im *end-to-end*-Modellieren von Assistenzabläufen [70, 30]. Hier wird davon ausgegangen, dass sogar die Struktur eines Assistenzmodells als neuronales Netzwerk gelernt werden kann, ohne dass dazu explizite Annahmen, wie wir sie oben ausgeführt haben, notwendig sind. Dieser Weg, Lernaufgaben für Assistenzsysteme zu lösen, benötigt große Corpora von entsprechend annotierten Kooperationsprotokollen (siehe die Beispiele oben; dort muss aber auch noch der Kooperationszweck annotiert werden sowie die Information, auf welchen Task sich einzelne Interaktionsschritte beziehen, und die Information welche Aktivität einer Problemlösung (zu welchem Task) gerade ausgeführt wird.

Solche Corpora sind leider noch nicht verfügbar. Es bleibt zu hoffen, dass sich – analog zu frei verfügbaren Tools wie RASA – eine open source community entwickelt, die nach einem standardisierten Schema Kooperationsprotokolle (und nicht nur Dialogtranskripte) annotiert und verfügbar macht, so dass wie im Bereich der Bilderkennung standardisierte Lösungen (vgl. z.B ImageNet [13]) für das *belief update* beim *dialogue state tracking* [72] und Strategien für domänenunabhängige Assistenzbedarfe entwickelt werden können, die sich für spezielle Domänen mit geeigneten zusätzlichen Trainingsdaten durch Neutraining adaptieren lassen (siehe Abschnitt 1.9.2). Wesentlich für die Etablierung eines Standards wird es sein, eine Methode zu entwickeln, mit deren Hilfe festgestellt werden kann, ob ein Corpus für die Anwendungsdomäne hinreichend repräsentativ ist, so dass ein darauf trainiertes Verfahren ausreichend generalisieren kann.

Die gesuchte Methode könnte in einem ersten Schritt darin bestehen, für alle Aktivitäten alle Assistenzbedarfe (siehe Tabelle 1.1 und Abbildung 1.7) und Formulierungen, wie sie in tatsächlichen Interaktionen vorkommen (können) dafür zu identifizieren. Sind alle Tripel aus Aktivität, Diagnose und Interaktionszweck mit Beispielen belegt, ist eine Erwartung dafür festgelegt, was Nutzer und Assistent in irgendeiner möglichen Situation sagen können. Diese Kenntnis kann sogar dazu genutzt werden, Assistenzabläufe mit natürlichsprachlichen Äußerungen in großem Umfang zu generieren, um nach dem Konzept der *data augmentation* ein erhobenes Corpus mit simulierten Daten zu vergrößern. Damit können dann *end-to-end*-Modelle — wie z.B. in [15] beschrieben — trainiert werden.

1.8 Wahrnehmung der Umgebung durch Sensorik

Für die Erstellung von Diagnosen (siehe Abb. 1.7), ob bzw. inwieweit die Ausführung einer Aktivität die erwarteten Effekte hervorgerufen hat, muss ein Assistenzsystem neben Information vom Nutzer (wie oben erläutert) auch Information über die aktuelle Situation der (physikalischen) Umgebung mit Hilfe von Sensoren erfassen, daraus den aktuellen Zustand des Tasks aktualisieren und eventuelle Abweichungen vom erwarteten Zustand feststellen. Die hierfür benötigten Algorithmen aus dem Bereich des Maschinellen Lernens sollen hier nicht weiter erörtert werden, sie werden an anderer Stelle in diesem Handbuch besprochen. Es ist eine Eigenschaft von Companion-Systemen ([9], S. 2f.), dass auch Klassifikatoren, meistens sogar für verschiedene Klassifikationsprobleme, zur Systemarchitektur gehören, wie dies auch in der Robotik der Fall ist.

1.9 Herausforderungen

Insgesamt haben wir gesehen, dass die Konzeption und Implementierung eines Assistenzsystems eine komplexe Aufgabe ist, zu deren Lösung verschiedene KI-Verfahren in geeigneter Weise zu einem Kontroll- und Beobachtungs-Algorithmus kombiniert werden müssen. Dies liegt in der Natur von Assistenzaufgaben, bei denen Interaktion in natürlicher Sprache, Aktionsplanung und -ausführung in einem komplexen Kontext zusammentreffen.

Damit sind alle Probleme und offenen Fragen, die bei automatischen Verstehen von (gesprochener) Sprache, bei der Interpretation von Information in einem (meist nur partiell beobachtbaren) Kontext und der Planung von Handlungen unter Unsicherheit auftreten, automatisch auch offene Fragen für ein allgemeines Konzept für Assistenzsysteme.

Es gibt also Forschungsbedarf in viele, und zum Teil ganz verschiedene Richtungen. Fortschritte dort werden auch Assistenzsysteme verbessern. Es geht aber nicht nur

darum, auf neue Ergebnisse aus anderen Teildisziplinen der KI zu warten, sondern auch im Bereich Assistenzsysteme gibt es offene Fragen, deren Beantwortung zu einer besseren theoretischen interdisziplinären und systematischen Fundierung von Companion-Technologie beitragen kann.

1.9.1 Intentionserkennung

Eine zentrale linguistische Herausforderung besteht in der Erkennung von Interaktions- und Kooperationszweck von sprachlichen Äußerungen der Nutzer (oft auch Intentionserkennung genannt). Es ist aktuell üblich, für das Verstehen von Äußerungen Corpora zu erheben, in denen einzelne Äußerungen mit dem wesentlichen domänenrelevanten Inhalt annotiert sind (z.B. [20]). Daraus lassen sich Information Extraction und Information Retrieval-Modelle lernen, die dem Umfang der Kapazität, Sprache zu interpretieren, abdecken. Es wird dabei außer Acht gelassen, dass aus Äußerungen auch Information darüber extrahiert werden kann, welche Intention ein Nutzer mit seiner Äußerung verfolgt. TP_3 ist auch hierfür ein Beispiel: „*Ok*“ drückt zunächst die Diskursfunktion einer Bestätigung aus, ist also ein sogenanntes *diskursives Mittel* [14], das im Kontext des aktuellen Taskzustands als positives Feedback zur in A_2 geleisteten Assistenz interpretiert werden kann. Später im sprachlichen Kontext von „*Öh (.) Ähm (?) Ja Ok*“ hingegen ist das Wort Bestandteil einer Phase, mit der Zögern ausgedrückt wird. Die Herausforderung besteht also erstens im Erheben von Datensätzen für diskursive Mittel und daraus resultierende Interaktionszwecke, zweitens in der Erstellung einer ausreichenden Liste diskursiver Mittel und drittens im Trainieren von Modellen (z.B. *word embeddings*) für die Prädiktion diskursiver Mittel und Interaktionszwecken zu beliebigen Äußerungen, in denen linguistische Cues (wie etwa „ok“) in einen größeren Kontext eingebettet sind. Diese Prädiktion kann dann zum *belief update* für das *state tracking* (siehe Abschnitt 1.7.1) herangezogen werden, um eine Disambiguierung des Interaktionszwecks im Kontext des bisherigen Kooperationsverlaufs zu erreichen.

1.9.2 Sprachverstehen in Assistenzkontexten

Die Verquickung verschiedener Domänen (siehe Abschnitt 1.4.1) in Äußerungen stellt nach wie vor eine der größten Herausforderungen für das Sprachverstehen dar, da ein Assistenzsystem — anders als etwa ein Chatbot (siehe z.B. [41]) den Kontext, also den Zustand aller Tasks (bzw. *task lineages*) exakt rekonstruieren muss, um zielgerichtet Assistenz leisten zu können. Dies ist (meist) nicht durch Vorhersage von Zeichensequenzen, die auf bisherige Zeichensequenzen folgen, zu leisten. Stattdessen müssen updates für (aus Sicht der Interaktion) latente Variablen (z.B. der Kooperationszweck oder Assistenzbedarf) explizit gemacht werden. Eine zukünftige

Lösung für diese Herausforderung könnte darin bestehen, dass Dialogcorpora aufgebaut werden, in denen Äußerungen für alle Domänen parallel annotiert werden anstatt nur den applikationsrelevanten Inhalt zu markieren (siehe Abschnitt 1.7.5). Ein geeignetes Annotationsschema könnte (in Erweiterung der Dialogbeispiele oben) folgende Dimensionen umfassen:

- ID
- Äußerung
- Kooperationszweck in Bezug auf Task ID
- Interaktionszweck in Bezug auf Task ID
- Strategie
- Task ID
- Problemlösung
- aktuelle Aktivität
- Phase der Aktivität
- Diagnose

Für die ersten beiden Äußerungen im ersten Beispiel für einen Dialog mit dem Küchenhelfer wäre (unter Auslassung des Wortlauts, um die Übersichtlichkeit zu wahren) folgendes Kooperationsprotokoll sinnvoll:

TP_1	neuer Task	Aktivierung	Aufforderung	t_1	Rezeptsuche	Zielfestlegung	Zielbildung	ok
	Fortführen t_1			t_1		Recommender-Aufruf		ok
A_2	Fortführen t_1	Information über ausgewählte Optionen zu TP_1	Filterassistenz	t_1		Mitteilung	Entscheidung über Auswahl	ok

Die zweite Zeile protokolliert eine Aktivität der Problemlösung, zu der es im Dialog keine Interaktion zwischen Nutzer und Küchenhelfer gibt, nämlich die eigentlich Suche nach Rezeptvorschlägen. Alle drei Zeilen (auch wenn sie zwei Äußerungen beinhalten) beziehen sich auf denselben durch TP_1 initiierten Task t_1. Mit solchen Annotationen können Modelle für das *belief update* trainiert werden. Die verschiedenen Annotationsdimensionen geben also eine umfangreiche Approximation des aktuellen Kontexts ab und erlauben es, komplexere Kooperationsabläufe zu analysieren und generieren als es mit aktuellen Tools wie RASA und vergleichbaren Systeme möglich ist.

1.9.3 Nicht modelliertes Handeln

Die Entwicklung eines Assistenzsystems erfordert, wie wir bisher erläutert haben, einen erheblichen Aufwand, um das notwendige Domänenwissen sogar für einfach strukturierte Kooperationsabläufe zu repräsentieren. Dies ist sogar unabhängig davon, ob die Verarbeitung des Wissens in einer Assistenzsituation mit Hilfe generischer Algorithmen (wie oben beschrieben) durchgeführt wird, oder ob bestimmte Abläufe a priori vom Dialogdesigner fixiert sind und vom Assistenzsystem Schritt für Schritt abgearbeitet werden. So funktionieren beispielsweise die *stories* in RASA: sie spezifizieren, welche Kooperationsabläufe überhaupt verarbeitet werden können.

Jeder der beiden Ansätze stößt an seine Grenze, wenn während der Kooperation Ereignisse stattfinden, zu denen das Assistenzsystem nicht über Wissen verfügt. In so einer Situation müsste das System seine Kompetenzen erweitern. Es bleibt der zukünftigen Forschung überlassen herauszufinden, inwieweit Verfahren des *Interactive Task Learning* [33] dafür verwendet werden können. Beinhaltet dies auch die Notwendigkeit, Kompetenzen der Kooperation und der Interaktions und des Sprachverstehens bis hin zum Vokabular eines Spracherkenners erweitern zu können? Kann ein Spracherkenner tatsächlich durch Vorsprechen neues Vokabular lernen? Wie also die notwendige Kompetenzerweiterung auf alle in einem Assistenzsystem integrierten KI-Verfahren weiterpropagiert werden kann, bleibt eine spannende Frage.

Ebenso spannend bleibt die Entwicklung einer geschlossenen interdisziplinären Theorie für Assistenzsysteme, die – in Fortschreibung der Ergebnisse zur Companion-Technologie – die für das Leisten von Assistenz in komplexen Szenarien notwendigen Kompetenzen in eine Systemarchitektur integriert und algorithmisch effektiv (und auch effizient) operationalisiert. Aktuelle kommerzielle Entwicklungen wie Amazon Echo, Sprachinterfaces für Smartphones, Sprachinterfaces im Auto und viele andere belegen aufgrund der hohen Nutzungszahlen, dass sowohl Anbieter als auch (potenzielle) Nutzer solcher Technologien großes Interesse haben, und dass in vielen Alltagsszenarien, wie z.B. in dem von Robert, aufgrund der Verfügbarkeit von hoher Rechenleistung in fast jedem Ort ganz neuartige Informationsdienstleistungen, nämlich Assistenz bei alltäglichen Aufgaben, unseren Alltag einfacher machen können.

Literatur

[1] James Allen, George Ferguson, and Amanda Stent. An architecture for more realistic conversational systems. In *Proceedings of the 6th International Conference on Intelligent User Interfaces*, IUI '01, pages 1–8, New York, NY, USA, 2001. ACM.

[2] James F. Allen and C. Raymond Perrault. Analyzing intention in utterances. *Artif. Intell.*, 15(3):143–178, 1980.

[3] Brandon Amos, Bartosz Ludwiczuk, and Mahadev Satyanarayanan. Openface: A general-purpose face recognition library with mobile applications. Technical report, CMU-CS-16-118, CMU School of Computer Science, 2016.

[4] LILIANA ARDISSONO, GUIDO BOELLA, and LEONARDO LESMO. A plan-based agent architecture for interpreting natural language dialogue. *International Journal of Human-Computer Studies*, 52(4):583 – 635, 2000.

[5] Gregor Behnke, Marvin Schiller, Matthias Kraus, Pascal Bercher, Mario Schmautz, Michael Dorna, Michael Dambier, Wolfgang Minker, Birte Glimm, and Susanne Biundo. Alice in diy wonderland or: Instructing novice users on how to use tools in diy projects. *AI Communications*, (Preprint):1–27, 2019.

[6] Pascal Bercher, Susanne Biundo, Thomas Geier, Thilo Hoernle, Florian Nothdurft, Felix Richter, and Bernd Schattenberg. Plan, repair, execute, explain - how planning helps to assemble your home theater. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*, 2014.

[7] Susanne Biundo, Daniel Höller, Bernd Schattenberg, and Pascal Bercher. Companion-technology: An overview. *KI - Künstliche Intelligenz*, 30(1):11–20, Feb 2016.

[8] Susanne Biundo and Andreas Wendemuth. Companion-technology for cognitive technical systems. *KI - Künstliche Intelligenz*, 30(1):71–75, 2016.

[9] Susanne Biundo and Andreas Wendemuth. *Companion Technology: A Paradigm Shift in Human-Technology Interaction*. Springer Publishing Company, Incorporated, 1st edition, 2017.

[10] Mihai Bâce, Sander Staal, and Andreas Bulling. Accurate and robust eye contact detection during everyday mobile device interactions, 2019.

[11] J Michael Crant. Proactive behavior in organizations. *Journal of management*, 26(3):435–462, 2000.

[12] Professor Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[14] Nicole Ehrmann. *Strukturen der Konzeptualisierung frühkindlicher Mehrsprachigkeit*. Dissertation, Universität Regensburg, 2016.

https://doi.org/10.1515/9783110218091-002

[15] Mihail Eric and Christopher D. Manning. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue, 2017.

[16] Kutluhan Erol, James A. Hendler, and Dana S. Nau. Complexity results for HTN planning. *Ann. Math. Artif. Intell.*, 18(1):69–93, 1996.

[17] Kutluhan Erol, Jun Lang, and Renato Levy. Designing agents from reusable components. In *Proceedings of the Fourth International Conference on Autonomous Agents, AGENTS 2000, Barcelona, Catalonia, Spain, June 3-7, 2000*, pages 76–77, 2000.

[18] Maria Fox and Derek Long. PDDL2.1: an extension to PDDL for expressing temporal planning domains. *J. Artif. Intell. Res.*, 20:61–124, 2003.

[19] Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In Jörg P. Müller, Michael J. Wooldridge, and Nicholas R. Jennings, editors, *Intelligent Agents III Agent Theories, Architectures, and Languages*, pages 21–35, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[20] Dayne Brian Freitag. *Machine Learning for Information Extraction in Informal Domains*. PhD thesis, Pittsburgh, PA, USA, 1999. AAI9937201.

[21] Alexander Frummet, David Elsweiler, and Bernd Ludwig. Detecting domain-specific information needs in conversational search dialogues. In Mehwish Alam, Valerio Basile, Felice Dell'Orletta, Malvina Nissim, and Nicole Novielli, editors, *Proceedings of the 3rd Workshop on Natural Language for Artificial Intelligence co-located with the 18th International Conference of the Italian Association for Artificial Intelligence (AIIA 2019), Rende, Italy, November 19th-22nd, 2019*, volume 2521 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019.

[22] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning and Acting*. Cambridge University Press, New York, NY, USA, 1st edition, 2016.

[23] Adam M Grant and Susan J Ashford. The dynamics of proactivity at work. *Research in organizational behavior*, 28:3–34, 2008.

[24] David Griol, Giuseppe Riccardi, and Emilio Sanchis. A statistical dialog manager for the LUNA project. In *INTERSPEECH 2009, 10th Annual Conference of the International Speech Communication Association, Brighton, United Kingdom, September 6-10, 2009*, pages 272–275, 2009.

[25] Barbara J. Grosz and Candace L. Sidner. Attention, intentions, and the structure of discourse. *Comput. Linguist.*, 12(3):175–204, July 1986.

[26] Nader Hanna and Deborah Richards. Speech act theory as an evaluation tool for human–agent communication. *Algorithms*, 12(4):1–17, 2019. Copyright the Author(s) 2019. Version archived for private and non-commercial use with the permission of the author/s and according to publisher conditions. For further rights please contact the publisher.

[27] Peter A. Heeman and James F. Allen. Speech repairs, intonational phrases, and discourse markers: Modeling speakers' utterances in spoken dialogue. *Comput. Linguist.*, 25(4):527–571, December 1999.

[28] Seongmin Heo and Jay H. Lee. Fault detection and classification using artificial

neural networks. *IFAC-PapersOnLine*, 51(18):470 – 475, 2018. 10th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2018.

[29] Jeffrey L Jenkins, Bonnie Brinton Anderson, Anthony Vance, C Brock Kirwan, and David Eargle. More harm than good? how messages that interrupt can make us vulnerable. *Information Systems Research*, 27(4):880–896, 2016.

[30] Zhuoxuan Jiang, Ziming Huang, Dong Sheng Li, and Xian-Ling Mao. Dialogact2vec: Towards end-to-end dialogue agent by multi-task representation learning, 2019.

[31] Matthias Kraus, Marvin Schiller, Gregor Behnke, Pascal Bercher, Susanne Biundo, Birte Glimm, and Wolfgang Minker. A multimodal dialogue framework for cloud-based companion systems. In *9th International Workshop on Spoken Dialogue System Technology*, pages 405–410. Springer, 2019.

[32] Akira Kurematsu, Youichi Akegami, Susanne Burger, Susanne Jekat, Brigitte Lause, Victoria MacLaren, Daniela Oppermann, and Tanja Schultz. VERBMOBIL dialogues: multifaced analysis. In *Sixth International Conference on Spoken Language Processing, ICSLP 2000 / INTERSPEECH 2000, Beijing, China, October 16-20, 2000*, pages 712–715. ISCA, 2000.

[33] John E. Laird, Kevin A. Gluck, John R. Anderson, Kenneth D. Forbus, Odest Chadwicke Jenkins, Christian Lebiere, Dario D. Salvucci, Matthias Scheutz, Andrea Lockerd Thomaz, J. Gregory Trafton, Robert E. Wray, Shiwali Mohan, and James R. Kirk. Interactive task learning. *IEEE Intelligent Systems*, 32:6–21, 2017.

[34] Sungjin Lee and Amanda Stent. Task lineages: Dialog state tracking for flexible interaction. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 11–21, Los Angeles, September 2016. Association for Computational Linguistics.

[35] Bernd Ludwig. Tracing actions helps in understanding interactions. In *SIGDIAL Workshop*, 2006.

[36] Bernd Ludwig. *Interaktion mit Assistenzsystemen*, pages 47–71. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[37] Bernd Ludwig. *Interaktive Assistenzsysteme*, pages 5–46. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[38] Bernd Ludwig. *Planbasierter Dialog*, pages 255–308. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[39] Bernd Ludwig. *Task-Analysen in der Mensch-Maschine-Interaktion*, pages 309–328. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[40] Colin Matheson, Massimo Poesio, and David Traum. Modelling grounding and discourse obligations using update rules. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, NAACL 2000, pages 1–8, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[41] Michael McTear, Zoraida Callejas, and David Griol. *The Conversational Inter-*

face: Talking to Smart Devices. Springer Publishing Company, Incorporated, 1st edition, 2016.

[42] Nikola Mrksic, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve J. Young. Neural belief tracker: Data-driven dialogue state tracking. *CoRR*, abs/1606.03777, 2016.

[43] Ramesh Nallapati. Semantic language models for topic detection and tracking. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Proceedings of the HLT-NAACL 2003 student research workshop-Volume 3*, pages 1–6. Association for Computational Linguistics, 2003.

[44] H. Niemann, A. Brietzmann, R. Mühlfeld, P. Regel, and G. Schukat. The speech understanding and dialog system evar. In Renato De Mori and Ching Y. Suen, editors, *New Systems and Architectures for Automatic Speech Recognition and Synthesis*, pages 271–302, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.

[45] Florian Nothdurft, Stefan Ultes, and Wolfgang Minker. Finding appropriate interaction strategies for proactive dialogue systems—an open quest. In *Proceedings of the 2nd European and the 5th Nordic Symposium on Multimodal Communication, August 6-8, 2014, Tartu, Estonia*, number 110, pages 73–80. Linköping University Electronic Press, 2015.

[46] Hyacinth S. Nwana. Software agents: an overview. *The Knowledge Engineering Review*, 11(3):205–244, 1996.

[47] Zhenhui Peng, Yunhwan Kwon, Jiaan Lu, Ziming Wu, and Xiaojuan Ma. Design and evaluation of service robot's proactivity in decision-making support process. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, page 98. ACM, 2019.

[48] Thies Pfeiffer and Ipke Wachsmuth. Multimodale blickbasierte Interaktion. *at - Automatisierungstechnik*, 61(11):770–776, 2013.

[49] Massimo Poesio and David R. Traum. Conversational actions and discourse situations. *Computational Intelligence*, 13:309–347, 1997.

[50] Tony Matthias Poitschke. *Blickbasierte Mensch-Maschine Interaktion im Automobil*. Dissertation, Technische Universität München, München, 2011.

[51] Pascal Poupart. *Partially Observable Markov Decision Processes*, pages 754–760. Springer US, Boston, MA, 2010.

[52] Matthew Purver, Jonathan Ginzburg, and Patrick Healey. On the means for clarification in dialogue. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*, 2001.

[53] Lawrence R. Rabiner. Readings in speech recognition. chapter A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[54] Raymond Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987.

[55] K. Ruhland, C. E. Peters, S. Andrist, J. B. Badler, N. I. Badler, M. Gleicher,

B. Mutlu, and R. McDonnell. A review of eye gaze in virtual agents, social robotics and hci: Behaviour generation, user interaction and perception. *Computer Graphics Forum*, 34(6):299–326, 2015.

[56] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.

[57] Richard Schaller, Morgan Harvey, and David Elsweiler. Detecting event visits in urban areas via smartphone gps data. In Maarten de Rijke, Tom Kenter, Arjen P. de Vries, ChengXiang Zhai, Franciska de Jong, Kira Radinsky, and Katja Hofmann, editors, *Advances in Information Retrieval*, pages 681–686, Cham, 2014. Springer International Publishing.

[58] Marvin Schiller, Gregor Behnke, Mario Schmautz, Pascal Bercher, Matthias Kraus, Michael Dorna, Wolfgang Minker, Birte Glimm, and Susanne Biundo. A paradigm for coupling procedural and conceptual knowledge in companion systems. In *2017 International Conference on Companion Technology (ICCT)*, pages 1–6. IEEE, 2017.

[59] Tony Sheeder and Jennifer Balogh. Say it like you mean it: Priming for structure in caller responses to a spoken dialog system. *International Journal of Speech Technology*, 6(2):103–111, 2003.

[60] Walter v. Hahn Susanne Jekat. Verbmobil: Foundations of speech-to-speech translation. chapter Multilingual Verbmobil-Dialogs: Experiments, Data Collection and Data Analysis, pages 575–582. Springer, 2000.

[61] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[62] Ryuichi Takanobu, Hanlin Zhu, and Minlie Huang. Guided dialog policy learning: Reward estimation for multi-domain task-oriented dialog. 2019.

[63] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[64] David R. Traum and James F. Allen. Discourse obligations in dialogue processing. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, ACL '94, page 1–8, USA, 1994. Association for Computational Linguistics.

[65] David R. Traum and Staffan Larsson. *The Information State Approach to Dialogue Management*, pages 325–353. Springer Netherlands, Dordrecht, 2003.

[66] Sebastian Varges, Silvia Quarteroni, Giuseppe Riccardi, Alexei Ivanov, and Pierluigi Roberti. Leveraging POMDPs trained with user simulations and rule-based dialogue management in a spoken dialogue system. In *Proceedings of the SIGDIAL 2009 Conference*, pages 156–159, London, UK, September 2009. Association for Computational Linguistics.

[67] H. Wandke. Assistance in human–machine interaction: a conceptual framework and a proposal for a taxonomy. *Theoretical Issues in Ergonomics Science*, 6(2):129–155, 2005.

[68] Edda Weigand. Tayler and Francis, 2017.

[69] Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. Latent intention dialogue models. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3732–3741, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[70] Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 438–449, Valencia, Spain, April 2017. Association for Computational Linguistics.

[71] Jason Williams. A belief tracking challenge task for spoken dialog systems. In *NAACL-HLT Workshop on Future directions and needs in the Spoken Dialog Community: Tools and Data (SDCTD 2012)*, pages 23–24, Montréal, Canada, June 2012. Association for Computational Linguistics.

[72] Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413, Metz, France, August 2013. Association for Computational Linguistics.

[73] Jason D. Williams, Kavosh Asadi, and Geoffrey Zweig. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 665–677, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[74] Jason D. Williams, Pascal Poupart, and Steve Young. *Partially Observable Markov Decision Processes with Continuous Observations for Dialogue Management*, pages 191–217. Springer Netherlands, Dordrecht, 2008.

[75] Jason D. Williams and Geoffrey Zweig. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *ArXiv*, abs/1606.01269, 2016.

[76] Michael Wooldridge. *An Introduction to MultiAgent Systems*. Wiley Publishing, 2nd edition, 2009.

[77] S. Young, M. Gašić, B. Thomson, and J. D. Williams. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179, May 2013.