Brandenburgische Technische Universität
Cottbus - Senftenberg

Lehrstuhl für Kommunikationstechnik
Fakultät 1

Masterarbeit

Kognitive Gerätesteuerung

zur Erlangung des akademischen Grades

Master of Science (M.Sc.)

Autor:	Peter Geßler geb. am 20. April 1989 in Potsdam
Matrikel-Nr.:	3002706
E-Mail:	gessler_peter@outlook.de
Prüfer:	Prof. Dr.-Ing. habil. Matthias Wolff
Zweitprüfer:	Dr.-Ing. Ronald Römer

Brandenburgische Technische Universität Cottbus - Senftenberg

Professur Kommunikationstechnik

Erklärung über die selbstständige Arbeit

Der Verfasser erklärt, dass er die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt hat Die aus fremden Quellen (einschließlich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind ausnahmslos als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

Ort, Datum: Cottbus, den 1. Dezember 2016 .....

(Unterschrift)

Kurzfassung

Um in Zukunft leistungsfähigere Maschinen zu konstruieren, die in der Interaktion mit ihrer Umwelt oder dem Benutzer stehen, müssen diese verschiedene kognitive Fähigkeiten emulieren und den zur Steuerung des Verhaltens programmierten Handlungsspielraum überschreiten können. Am Lehrstuhl Kommunikationstechnik der Brandenburgischen Technischen Universität werden seit einiger Zeit Methoden und Ansätze zur Konstruktion dieser sogenannten *technischen dynamischen kognitiven Systeme* erforscht. Der Fokus liegt dabei auf der intuitiven Steuerung der Maschine durch Sprachäußerungen eines Benutzers und der Verarbeitung von semantischen Datenstrukturen. Ein technisches dynamisches kognitives System verwendet diese in seiner Verhaltenssteuerungskomponente, um beispielsweise Schlussfolgerungen zu treffen, Handlungen auszuführen oder das weitere Systemverhalten zu planen. Die vorliegende Arbeit beschäftigt sich erstmals mit der Verwendung von beschrifteten gewichteten Merkmal-Werte-Relations Graphen (semantische Datenstruktur) innerhalb einer Verhaltenssteuerungskomponente, welche für die Steuerung eines Heizungssystems verantwortlich ist. Der Aufbau des technischen dynamischen kognitiven Systems orientierte sich an dem systemtheoretischen Modell des *doppelten kognitiven Kreises*, das in einer Forschungssitzung am Lehrstuhl Kommunikationstechnik entstand. Neben der ausführlichen Beschreibung von beschrifteten gewichteten Merkmal-Werte-Relations Graphen sind in dieser Arbeit ebenfalls die Operationen *bgMWR-Hinzufügen*, *bgMWR-Löschen* und *bgMWR-Vereinheitlichung* mathematisch definiert worden. Des Weiteren ist mit der prototypischen Umsetzung der Verhaltenssteuerungskomponente erstmals ein System beschrieben, welches ausschließlich durch die zuvor verarbeiteten beschrifteten gewichteten Merkmal-Werte-Relations Graphen sein Systemverhalten anpasst und auf die Umwelt einwirkt. Die Implementierung des Prototyps erfolgte in das bereits bestehende *CSL-Projekt* des Lehrstuhls Kommunikationstechnik mit der Programmiersprache Java. Für die weitere Forschung auf dem Gebiet der technischen dynamischen kognitiven Systeme wurden abschließend weiterführende Problemstellungen kritisch betrachtet und verschiedene Lösungsansätze diskutiert.

Formelzeichen, Symbole und Abkürzungen

Die Formelzeichen und Symbole beziehen sich ausschließlich auf die in dieser Arbeit aufgestellten Definition und Operationen.

Formelzeichen - Graphen

A	Menge mit Beschriftungen	Q	Menge ähnlichster Pfade
c	Konfidenzwert	q	Präfix
$deg_<(v)$	Prägrad des Knotens v	R	Pfadmenge
$deg_>(v)$	Postgrad des Knotens v	r	Pfade
E	Kantenmenge	$R^>$	Menge maximaler Pfade
G	gerichteter Graph	$\tilde{r}$	Pfadrest
Γ	Nachbarschaft des Knotens v	$\mathbb{R}^+$	Menge der positiv reellen Zahlen
$\Gamma_<(v)$	Vorgänger des Knotens v	T	gerichteter, zusammenhängender, kreisfreier Graph
$\Gamma_>(v)$	Nachfolger des Knotens v	uv	Kante vom Knoten u zum Knoten v
k	diskreter Zeitpunkt	V	Knotenmenge
κ	Gewichtsfunktion	v	Knoten
λ	Abbildungsfunktion	v_0	Startknoten
$\mathcal{M}$	linksseitige Relationselemente	v_n	bestimmter Knoten
M	Merkmalmenge	W	atomare Wertemenge
m	Merkmal		
m_0	Merkmal der Wurzel		
$\mathbb{N}$	Menge der natürlichen Zahlen		
ν	Beschriftungsfunktion		

Formelzeichen - Semantik

$\mathcal{A}$	Vorgeschichte	$\mathcal{G}$	Grammatik
a	semantischer Anker	M'	Teilbedeutung
$\mathcal{B}$	Verhalten des Objekts	S	semantisches Schema
$\mathcal{C}$	Konsequenz	$\mathcal{T}$	Transduktor
D	Domäne	$\mathcal{U}$	Menge der Sprachäußerungen

Symbole

$\cup$	Vereinigung auf Mengen	$\mapsto$	bildet ab auf
$\cap$	Schnitt auf Mengen	$\oplus$	bgMWR-Hinzufügen
$\setminus$	Differenz auf Mengen	$\ominus$	bgMWR-Löschen
$V_{\mid}$	Mengenrestriktion	$\uplus$	bgMWR-Vereinheitlichung
$\times$	kartesisches Produkt	$\bigoplus$	Summe der Automatenalgebra

Wortabkürzungen

ASR	Automatic Speech Recognition
A-E Paar	Aktions-Erwartungshaltungs Paar
bMWR	beschriftete Merkmal-Werte-Relation
bgMWR	beschriftete gewichtete Merkmal-Werte-Relation
CSL	Cognitive System Lab
Cyc	Cycorp
EPG	Electronic Program Guide
eBNF	erweiterte Backus Naur Form
GUI	Graphical User Interface
MDP	Markov Decision Process
MWP	Merkmal Werte Permutation
MWR	Merkmal-Werte-Relation
NER	Named Entity Recognizer
POMDP	Partially Observable Markov Decision Process
REL	Relation Extraction
T-E Paar	Teilbedeutungs-Erwartungshaltungs Paar
UCUI	Universal Cognitive User Interface

I. Inhaltsverzeichnis

1. Einführung

> Neben den traditionellen Kulturtechniken Lesen, Schreiben und Rechnen tritt als eine neue vierte Kulturtechnik die Handhabung computergestützter Informations- und Datenverarbeitung. [34, W. Zimmerli, 1988]

Computerbasierte Systeme sind heutzutage nicht mehr aus dem Alltag wegzudenken. Selbst wer versucht, der fortschreitenden Digitalisierung zu entgehen, vertraut sich indirekt computergestützten Produktions-, Versorgungs-, Verkehrssystemen und so weiter an. Zur Verbesserung von Arbeits-, Produktions- und anderen Prozessen sind in Zukunft jedoch technologische Lösungen nötig, die Aufgaben nicht nur abarbeiten beziehungsweise sich in einer Umgebung nicht nur verhalten, sondern auf diese sinnvoll einwirken. Sie erfassen mit Sensoren Signale aus der Umwelt und können durch ihre Verhaltenssteuerung und ihre Aktuatoren mit dieser sinnvoll interagieren beziehungsweise operieren. Dadurch sind sie in der Lage, auch nicht vorausgeplante Probleme adäquat zu lösen. Es gilt, mit der Verhaltenssteuerung eine Brücke zwischen virtueller und dinglicher Welt herzustellen, wodurch eine wechselseitige Synchronisation zwischen dem digitalem Modell und der physischen Realität möglich ist. Zur Umsetzung dieser Anforderungen benötigt ein System die *kognitiven Fähigkeiten* (vgl. [8], [31], [30]):

1. Wahrnehmung - Aufnahme, Analyse und Interpretation von Signalen der Umwelt,
2. Lernen - Erwerb und Adaption von Wissen,
3. Planung unter Unsicherheit - Auswahl einer Strategie ohne Kenntnis des konkreten Umweltzustands und der Auswirkungen auf die Umgebung,
4. Schlussfolgern unter Unsicherheit - Ausführung einer Aktion, ohne den exakten Umweltweltzustand und die Auswirkungen zu kennen,
5. Handeln - auf die Umwelt über Aktuatoren sinnvoll einwirken,
6. Vorstellungskraft - Generierung von nicht programmierten Systemzuständen und Aktionen, die schneller zum Handlungsziel führen und
7. Bewältigungsverhalten (engl. *Coping*) - Überwindung einer Zugangsbarriere.

Ausgehend von dem Modell des *Software-Agenten* lässt sich sagen, dass die Handlungen eines *kognitiven Agenten* in Zukunft von den wahrgenommenen Signalen, dem erlernten Wissen, seiner Planung und den daraus getroffenen Schlussfolgerungen abhängig sind (vgl. [7]). Des Weiteren kann er durch Vorstellungskraft und Bewältigungsverhalten seinen initialen Handlungsspielraum überschreiten (vgl. [30]), um Probleme „intelligent“ zu lösen.

Bereits heute sind technische Produkte und Softwaredienste mit Algorithmen im Umlauf, die kognitive Fähigkeiten emulieren. Ein Hauptproblem besteht allerdings immer noch darin, dass sie den programmierten Handlungsspielraum bei der Steuerung ihres Verhaltens nicht überschreiten können, um ein Handlungsziel zu erreichen. Das stellt aus Sicht führender Technologieunternehmen ein großes Defizit in der Entwicklung neuer Produkte und Dienste dar [22]. Nach Ansicht des Lehrstuhls Kommunikationstechnik der Brandenburgischen Technischen Universität und Forschungspartnern ist dies erst durch die Entwicklung von „intelligenten Maschinen“[1] möglich, die semantische Datenstrukturen zur Informationsverarbeitung verwenden.
Eine technische Umsetzung erfordert zunächst jedoch weitere grundlegende Erkenntnisse bezüglich eines technischen dynamischen Systems, das semantische Datenstrukturen verarbeiten kann und niedrige kognitive Fähigkeiten wie beispielsweise Wahrnehmung, Lernen, Planung, Schlussfolgerung sowie Handeln unter deterministischen Bedingungen emuliert. Die Struktur eines konventionellen kognitiven Systems ist in Abbildung 1 dargestellt.

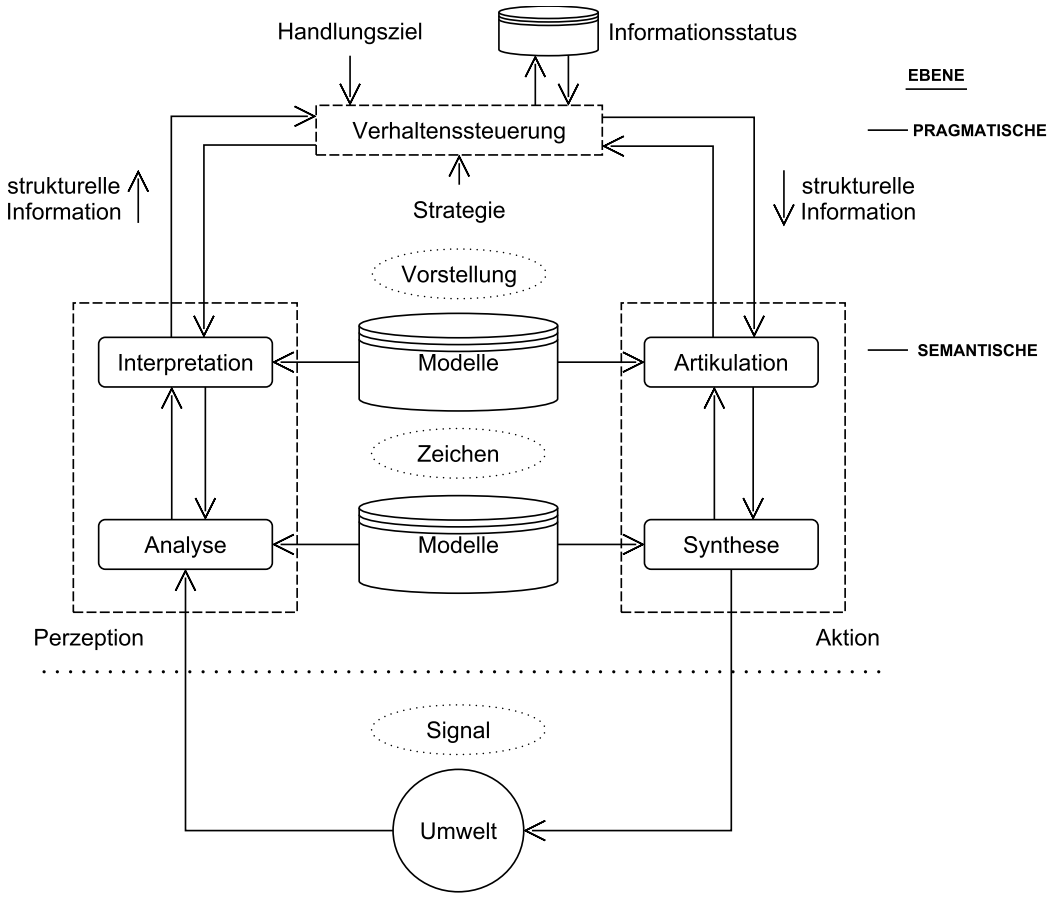


Abbildung 1: Erweiterter Perzeptions-Aktions-Zyklus in Anlehnung an [21, Abb. 1b].

[1] Computergestützte Systeme, welche die aufgelisteten kognitiven Fähigkeiten emulieren.

Aus systemtheoretischer Sicht ähnelt das Modell einem Regelkreis. Während sich die Informationsverarbeitung in der klassischen Regelungstechnik jedoch auf die Signalebene beschränkt, findet sie bei einem System mit kognitiven Fähigkeiten auf der semantischen und pragmatischen Ebene statt (vgl. [9], [20]). In der Literatur verwendet man deswegen den Begriff *Perzeptions-Aktions-Zyklus* oder *kybernetischer Zyklus* (vgl. [8]). Das beschriebene Modell eignet sich als Ansatz zur Realisierung von kognitiven Nutzerschnittstellen nach [31] oder autonomen Systemen wie zum Beispiel *Shannons Maus Theseus* (vgl. [21]). In der vorliegenden Arbeit wird das Modell aus der Abbildung 1 um einen inneren Perzeptions-Aktions-Zyklus erweitert. Der äußere Zyklus realisiert die kognitive Nutzerschnittstelle, während der innere Zyklus in Abbildung 2 die Kommunikation mit den inneren physischen Komponenten einer konkreten Maschine ermöglicht. Das gesamte System bezeichnen wir im Folgenden als *Assistenzsystem mit kognitiver Nutzerschnittstelle*, die ebenfalls eine Kategorie von technischen dynamischen kognitiven Systemen darstellen. Zentraler Bestandteil des Systems ist die *Verhaltenssteuerung*, welche nach einer Strategie versucht, ein vordefiniertes Handlungsziel zu erreichen.

Diese Arbeit beschäftigt sich mit der prototypischen Umsetzung der Verhaltenssteuerung in einem konkreten Assistenzsystem mit kognitiver Nutzerschnittstelle, wobei der Informationsfluss mit beschrifteten gewichteten Merkmal-Werte-Relationen (bgMWR) erfolgt.

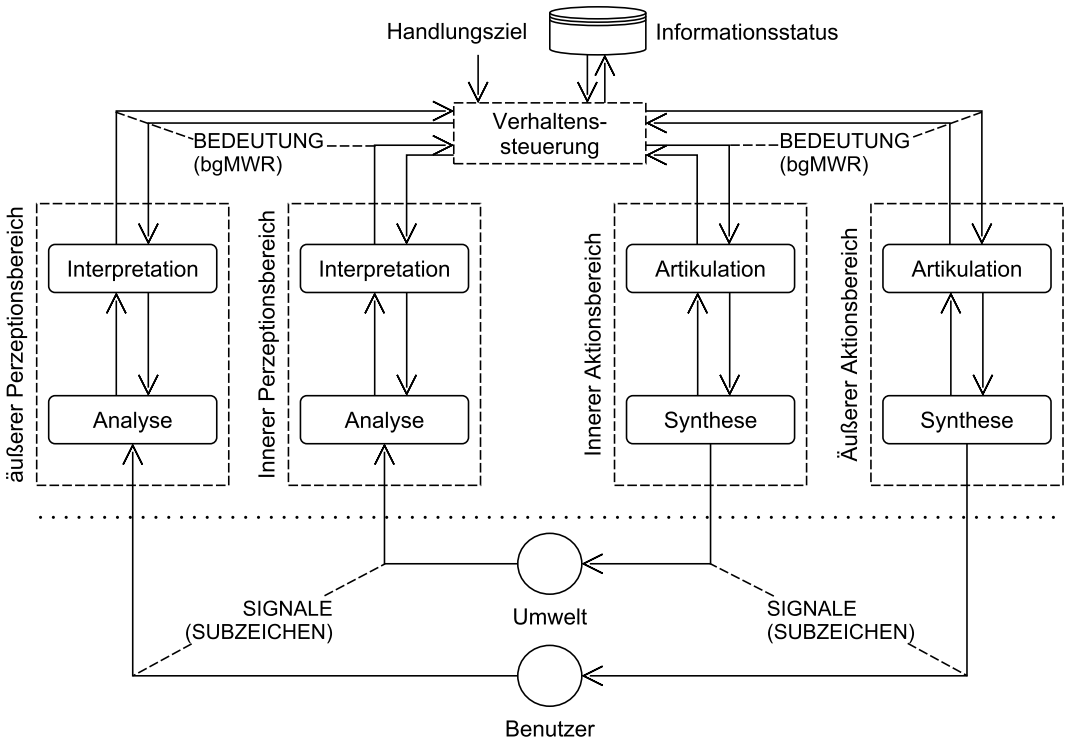


Abbildung 2: Blockdiagramm eines erweiterten doppelten Perzeptions-Aktions-Zyklus. Der Begriff *Bedeutung* ist in Abschnitt 2.3 erläutert.

Die entwickelte Verhaltenssteuerung besteht aus einer Dialogsteuerung (kognitive Nutzerschnittstelle) zur Regulierung der Dialogführung und einer Heizungssteuerung zur Regulierung der Raumtemperatur. Die Dialogsteuerung verarbeitet Spracheingaben, die als beschriftete gewichtete Merkmal-Werte-Relation-Zeichenketten vorliegen und generiert eine beschriftete gewichtete Merkmal-Werte-Relations-Zeichenkette als Ausgabe in Abhängigkeit von einem *Weltmodell* sowie des aktuellen Informationsstatus, der sich aus dem Dialog- und Heizungsstatus ergibt. Damit die Heizungssteuerung eine Systemaktion ausführen kann, benötigt sie Informationen wie zum Beispiel Ausführungszeitpunkt oder Temperaturangabe. Das System fragt den Benutzer bei einer „unvollständigen Nutzereingabe“ (aus Sicht des Systems) nach den fehlenden Informationen. Handelt es sich im nächsten Dialogschritt um eine beschriftete gewichtete Merkmal-Werte-Relation, die Informationen zur letzten unvollständigen Nutzereingabe beinhaltet, vereinheitlicht die Verhaltenssteuerung beide Teilinformationen miteinander. Kann jedoch keine Übereinstimmung zwischen der aktuellen Information und dem Wissen der Verhaltenssteuerung festgestellt werden, verarbeitet das System die empfangene bgMWR als neues Nutzerziel. Da es sich bei der ausgewählten Anwendung um eine Maschinensteuerung handelt, steht der Arbeitstitel *Kognitive Gerätesteuerung* als Synonym für das entwickelte Assistenzsystem mit kognitiver Nutzerschnittstelle.

Am Lehrstuhl Kommunikationstechnik der Brandenburgischen Technischen Universität existieren bereits Algorithmen zur Sprach-Semantik-Übersetzung und Semantik-Sprach-Übersetzung sowie Spracherkennung und -synthese. Die sprachliche Interaktion des entwickelten Prototyps ist deshalb nur von und bis zur syntaktischen Ebene ausgeführt. Die von der Verhaltenssteuerung zu verarbeitenden semantischen Datenstrukturen sind in dieser Arbeit beschriftete gewichtete Merkmal-Werte-Relations Graphen, wobei alle Gewichte den numerischen Wert 1 haben. Die kognitiven Fähigkeiten Vorstellungskraft und Bewältigungsverhalten sind nicht näher betrachtet worden.

Kapitel 2 beschäftigt sich zunächst mit dem Stand der Technik sowie den Grundlagen der entwickelten kognitiven Gerätesteuerung. Im Vordergrund steht dabei die Repräsentation beziehungsweise Verarbeitung von Semantik durch beschriftete gewichtete Merkmal-Werte-Relationen und eine Erläuterung der einzelnen Bereiche unterhalb der Verhaltenssteuerung in Abbildung 2. Kapitel 3 befasst sich mit den verwendeten Modellen, welche zur dynamischen Anpassung der Dialogführung notwendig sind. In Kapitel 4 sind die Systemarchitektur sowie softwaretechnische Umsetzung der entwickelten kognitiven Gerätesteuerung beschrieben. Abschließend gibt Kapitel 5 einen Ausblick auf die Weiterentwicklung kognitiver dynamischer Systeme mit kognitiver Nutzerschnittstelle inklusive damit verbundener Fragestellungen und Lösungsansätze.

2. Theoretische Grundlagen

Dieses Kapitel beschreibt im ersten Abschnitt den Stand der Technik von Assistenzsystemen mit Sprachdialogsystemen als kognitive Nutzerschnittstelle hinsichtlich der verwendeten Dialogsteuerungsverfahren und einer konkreten Anwendung. Anschließend beschäftigt sich der zweite Abschnitt mit der mathematischen Struktur und Darstellung eines beschrifteten gewichteten Merkmal-Werte-Relations-Graphen sowie Operationen, die zur Modifizierung dienen. Der dritte Abschnitt beschreibt im ersten Teil die Stellung eines beschriften gewichteten Merkmal-Werte-Relations-Graphen innerhalb des Systems. Der zweite Teil behandelt die Hin- und Rückübersetzung einer Zeichenkette in eine beschriftete gewichtete Merkmal-Werte-Relations-Zeichenkette.

2.1. Stand der Technik - Sprachdialogsysteme in Assistenzsystemen

Ein Sprachdialogsystem besteht im Allgemeinen aus miteinander operierenden Komponenten, welche für die Spracherkennung, Sprach-Semantik-Übersetzung, Dialogsteuerung, Semantik-Sprach-Übersetzung und Sprachsynthese zuständig sind. Abbildung 3 zeigt den Informationsfluss in einem prototypischen Sprachdialogsystem nach [14]. Neben dem klassischen Informationsfluss (durchgehende Linien) können zusätzliche Kontextinformationen (gestrichelte Linien) an die Komponenten übermittelt werden.

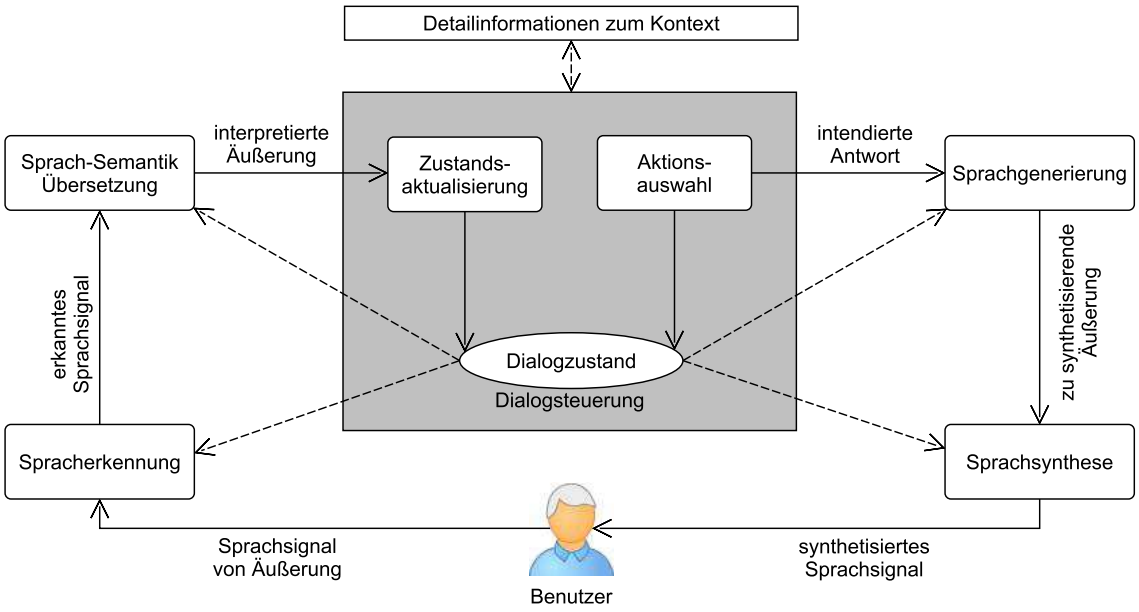


Abbildung 3: Komponenten und Informationsfluss in einem prototypischen Sprachdialogsystem mit *Blackboard-Architektur* nach [14].

Die Komponente zur Dialogsteuerung übernimmt mit der Repräsentation des aktuellen Dialogstatus, dessen Aktualisierung und der Auswahl einer Aktion drei zentrale Aufgaben im Sprachdialogsystem. Der Dialogzustand beinhaltet alle für das System relevanten Informationen wie zum Beispiel den bisherigen Dialogverlauf, den momentanen Konversationsstatus oder Informationen über den Status einer nicht abgeschlossenen Aufgabe. Basierend auf dem aktualisierten Dialogzustand trifft das Dialogsystem eine Entscheidung, die der Auswahl einer Aktion entspricht. Das kann entweder eine *Dialogaktion* zur Kommunikation mit dem Benutzer, eine *Systemaktion* zur Kommunikation mit einem weiteren externen System[2] oder die Kombination von Dialog- und Systemaktion sein. Zur Auswahl einer bestimmten Aktion, in Abhängigkeit des aktualisierten Dialogzustands, existieren verschiedene Dialogsteuerungsverfahren. Im Folgenden werden die bekanntesten vorgestellt. Ein besonderer Fokus liegt auf der planbasierten und statistischen Dialogsteuerung. Anschließend betrachten wir den deterministischen und probabilistischen Ansatz des Dialogsystems *End-to-End Dialog System for TV Program Discovery* der Firma *Nuance Communications Incorporated*. Das System folgt dabei dem Prinzip des *Konversationsfokus* nach [5] und versucht dem Benutzer verschiedene Handlungsalternativen anzubieten, wenn dies möglich ist. Das Dialogsystem der entwickelten Verhaltenssteuerung verwendet einen ähnlichen Ansatz hinsichtlich der Semantikrepräsentation und Verarbeitung.

2.1.1. Verfahren zur Dialogsteuerung

Zustandsbasierte Dialogsteuerung

Die zustandsbasierte Dialogsteuerung erfolgt mit Hilfe deterministischer endlicher Zustandsautomaten und wird nach [6] überwiegend in der Automobilindustrie verwendet. Ein endlicher Zustandsautomat ist nach [14] formal als Tupel $(\mathcal{Z}, \Sigma, \delta, z_0, \mathcal{F})$ definiert. $\mathcal{Z}$ ist eine endliche Menge von Zuständen, die das System einnehmen kann. Σ beschreibt alle vom System zu akzeptierenden Eingaben (in unserem Fall Spracheingaben des Benutzers). Durch die Übergangsfunktion $\delta : \mathcal{Z} \times \Sigma \to \mathcal{Z}$ bildet jedes *(Zustand, Eingabe)-Paar* auf einen Folgezustand ab. Bei der Initialisierung nimmt der Automat den Zustand z_0 ein und durchläuft ihn anschließend bis zu einem Endzustand $z \in \mathcal{F}$. Der Aufbau eines endlichen Zustandsautomaten zur Dialogsteuerung ist beispielhaft in Abbildung 4 dargestellt. Endliche Zustandsautomaten bieten einen einfachen Ansatz zur Erstellung einer Dialogsteuerung. Die Flexibilität zur Laufzeit ist jedoch stark eingeschränkt, weil alle Zustandsübergänge bereits während der Entwicklung festgelegt werden. In der Automobilindustrie ist das Verfahren jedoch durch die vollständige Testbarkeit des Zustandsautomaten und der minimalistischen Ausführung zur Laufzeit noch immer anerkannt.

[2] Das externe System ist in dieser Arbeit eine softwaretechnische Realisierung zur Steuerung der Heizung.

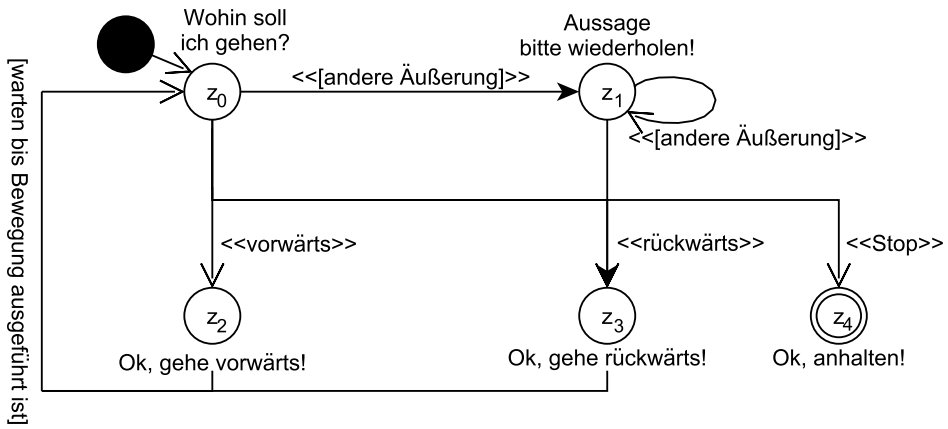


Abbildung 4: Beispiel eines endlichen Zustandsautomaten für die Dialogsteuerung. Das System führt bei den Sprachäußerungen „vorwärts“, „rückwärts“ sowie „stop“ eine Dialog- und Systemaktion aus. Bei anderen Äußerungen bittet das System um eine Wiederholung.

Regelbasierte Dialogsteuerung

Bei der regelbasierten Dialogsteuerung besteht der Dialogzustand aus einer endlichen Anzahl von Slots (*slot-filling* oder *frame-based*, vgl. [14],[17]). Diese sind beim Start des Systems leer und werden durch die Sprachäußerungen des Nutzers anschließend mit Werten gefüllt. Das System überprüft in Abhängigkeit der Slot-Zustände eine Anzahl von vordefinierten Regeln, um die nächste Aktion – zum Beispiel Rückfrage bezüglich eines nicht besetzten Slots im aktuellen Dialogzustand – auszuwählen. Nach der Ermittlung aller Slot-Werte-Paare wechselt das System entweder in einen anderen Dialogzustand oder beendet den Dialog nach der Ausführung einer Aktion. Im Gegensatz zur zustandsbasierten Dialogsteuerung kann mit dem *slot-filling* Ansatz eine größere Menge von Werten erfasst werden, die zur Überführung in einen Folgezustand notwendig sind. Damit eignet sich dieser Ansatz vor allem für Expertensysteme. Eine Änderung des Nutzerziels kann das System, wegen der reihenfolgebasierten Erfassung von Werten zu den Slots, jedoch nicht erkennen.

Planbasierte Dialogsteuerung

Mit einer planbasierten Dialogsteuerung versucht der Agent, das Handlungsziel des Nutzers anhand seines verbalen Verhaltens abzuleiten. Der Informationszustand ist durch ein *Belief, Desire and Intentions Modell* dargestellt, das die Annahmen, Wünsche und Intentionen des Nutzers umfasst. In Abhängigkeit des bisherigen Dialogverlaufs versucht die Dialogsteuerung einen Plan[3] zu erkennen, der das Nutzerverhalten am plausibelsten in

[3]Der Plan ist eine Sequenz von zu erwartenden Sprachäußerungen und Aktionen, die eine Handlung auslösen. Alle Pläne sind statisch im System hinterlegt.

der entsprechenden Domäne beschreibt. Die Auswahl einer Aktion ist vom ausgewählten Plan abhängig. Der Ansatz eignet sich nach [6] vor allem für Aufgaben, bei denen viele Vorbedingungen zum Auslösen einer Aktion erfüllt werden müssen oder Pläne miteinander konkurrieren. In [25] erfolgte die Umsetzung der planbasierten Dialogsteuerung mit Hilfe des *information state* Ansatzes. Dieser ermöglicht die Erkennung eines Nutzerziels durch den Zugriff auf die gesamte Wissensbasis des Agenten, welche zum Beispiel als *Wissensgraph* hinterlegt ist. Der Informationszustand des Systems besteht zum Beispiel aus einer endlichen Anzahl von Kellerspeichern, die den bisherigen Dialogverlauf, den momentanen Konversationsstatus sowie Informationen über den Status einer nicht abgeschlossenen Aufgabe des Systems repräsentieren. Erfolgt vom Nutzer eine neue Sprachäußerung, versucht das System durch Aktualisierungsregeln den bisherigen Konversationsstatus anzupassen. Gelingt dies nicht, geht das System von einem neuen Nutzerziel aus und legt die dazugehörige Information als oberstes Element auf den entsprechenden Kellerspeicher. Der Informationszustand beinhaltet somit alle Ein- und Ausgabedaten, die während des Dialogverlaufs entstehen. Das System wählt nach jedem Dialogschritt die Pläne aus einer Planungskomponente aus, welche mit dem bisherigen Dialogverlauf korrespondieren.

Statistische Dialogsteuerung

Die bisher vorgestellten Verfahren basieren auf der Annahme, dass eine ideale Erfassung der Benutzerintention zur getätigten Sprachäußerung durch den Spracherkennungs- und Sprach-Semantik-Übersetzungsprozess erfolgt. Das ist in der Realität jedoch nicht der Fall. Des Weiteren weicht das tatsächliche Nutzerverhalten oft von dem vorgegebenen beziehungsweise vermuteten Verhalten (bisher durch Zustandsautomat, Regeln oder Pläne dargestellt) ab. Statistische Ansätze basieren deshalb auf der Idee, die Dialogstrategie automatisch durch Trainingsdaten und Adaption zu optimieren. Ein möglicher Ansatz zum Erlernen der Dialogstrategie ist das *Verstärkungslernen* (engl. *reinforcement learning*).

Zur Laufzeit ändert der Agent im Allgemeinen durch das Ausführen einer Aktion den Zustand seiner Umwelt (in unserem Fall den Zustand des Benutzers). Nach jeder Aktion kann der Agent den neuen Benutzerzustand (Ergebnis der ausgeführten Aktion) beobachten. Die Ausführung einer bestimmten Aktion a in einem Zustand z ist dabei mit der Belohnung r verbunden. Das kann ein positiver oder negativer numerischer Wert sein, den der Agent in Abhängigkeit zur Relation des vorgegebenen Handlungsziels erhält. Während des Lernprozesses versucht der Agent, die auszuführende Aktion bezüglich des Zustands zu finden, welche eine Maximierung der Gesamtbelohnung zur Folge hat. Zur Umsetzung des Verfahrens nutzt man zum Beispiel den *Markov-Entscheidungsprozess* (engl. *Markov Decision Proceess*, MDP). Dieser ist als Tupel $(\mathcal{Z}, \mathcal{A}, T, R)$ definiert, wobei $\mathcal{Z}$ die Menge der möglichen Zustände und $\mathcal{A}$ die Menge ausführbarer Aktionen ist. Die

Übergangsfunktion T beinhaltet alle Wahrscheinlichkeiten $P(z'|z,a)$, die das Erreichen eines Folgezustands z' vom Zustand z durch das Ausführen einer Aktion a beschreiben. Mit dem Ausführen einer Aktion a im Zustand z erhält der Agent eine Belohnung r, die in der Belohnungsfunktion R festgelegt ist. Damit der Agent einen gegebenen Markov-Entscheidungsprozess optimal löst, ermittelt er in der Lernphase die Strategie $\pi : \mathcal{Z} \rightarrow \mathcal{A}$. In dieser ist jedem Zustand diejenige Aktion zugeordnet, bei der eine Maximierung von noch zu erwartenden Belohnungen bis zum Erreichen des Handlungsziels vorliegt. Der Markov-Entscheidungsprozess basiert jedoch ebenfalls auf der Annahme, dass eine ideale Erfassung des Umweltzustands vorliegt. Zur Beachtung von Unsicherheit bei der Erkennung kann der MDP zu einem partiell beobachtbaren Markov-Entscheidungsprozess (engl. *Partially Observable Markov Decision Process*, POMDP) erweitert werden. Dieser ist durch das Tupel $(\mathcal{Z}, \mathcal{A}, T, R, \mathcal{O}, Z)$ definiert, wobei der Beobachtungsraum $\mathcal{O}$ alle möglichen Umweltzustände beinhaltet. Der Agent verarbeitet anschließend eine endliche Menge von möglichen Umweltzuständen (zum Beispiel eine Liste der n-besten Beobachtungen). Die Wahrscheinlichkeit $P(o|z)$ des Umweltzustands im aktuellen Zustand ist durch die Funktion Z definiert. Eine detaillierte Betrachtung des POMDP-Verfahrens findet in [14], [31] und [33] statt.

Verschiedene Studien (vgl. [32]) haben gezeigt, dass statistische Ansätze zur automatischen Optimierung der Dialogstrategie bessere Ergebnisse[4] erzielen als händisch erstellte Dialogmodelle. Ein entscheidendes Problem ist jedoch das Training der statistischen Modelle. Diese benötigen im Allgemeinen eine große Anzahl von Daten, um die Modellparameter korrekt zu schätzen. Das Akquirieren der Daten ist oft sehr teuer und zeitaufwendig. Des Weiteren müssen die verwendeten Lernverfahren oftmals mit einer Approximation arbeiten, damit die Ausführung in akzeptabler Zeit erfolgt. Das betrifft vor allem die begrenzte Suche im Zustands- und Aktionsraum. Digitale Assistenten mit kognitiven Fähigkeiten besitzen jedoch meistens einen sehr komplexen Zustands-Aktionsraum, der zum Beispiel das parallele Bearbeiten von Aufgaben in Abhängigkeit der dynamischen Umwelt darstellt. Eine Beschränkung des Suchraums würde eventuell bessere Aktionen zur aktuellen Umweltsituation ausschließen. Nach *P. Lison* ([14]) kann dieses Problem nur durch einen hybriden Ansatz mit statistischer Modellierung und hinterlegtem Expertenwissen gelöst werden. Eine Diskussion der damit verbundenen Vor- und Nachteile ist jedoch nicht Gegenstand dieser Arbeit.

[4] Nach [14] umfasst dies objektive und subjektive Metriken.

2.1.2. Ein Dialogsystem zur Auswahl des TV-Programms

In den Beiträgen [18] und [19] präsentiert die Firma *Nuance Communications Inc.* ein Sprachdialogsystem zur Auswahl des TV-Programms. In diesem werden verschiedene Technologieverfahren wie Spracherkennung, Sprach-Semantik-Übersetzung, Dialogsteuerung mit relationalen Datenstrukturen, Schlussfolgerung auf Wissensgraphen und Sprachgenerierung eingesetzt. Das System basiert auf einer *Client-Server-* sowie *Nabe-Speichen-*Architektur (siehe Abbildung 5) und verwendet den *Information State Ansatz* nach [25]. Die Beschreibung des Dialogzustands erfolgt mit einer Wahrscheinlichkeitsverteilung über eine endliche Anzahl von möglichen Zuständen der Kellerspeicher.

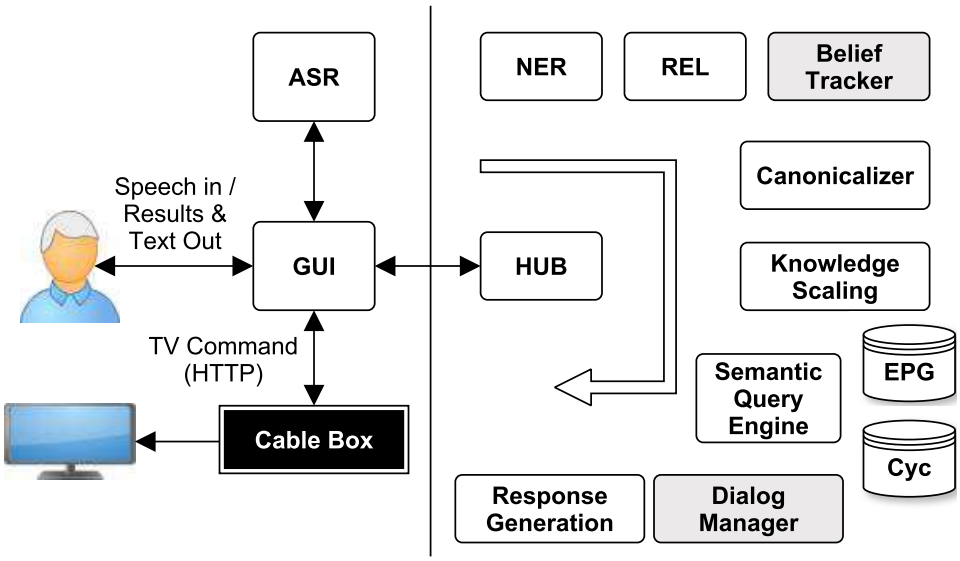


Abbildung 5: Aufbau des Sprachdialogsystems. Die Abbildung ist aus [19] rekonstruiert und repräsentiert die einzelnen Komponenten des Systems.

Die Komponenten des Clients sind für die Erfassung sowie Konvertierung der erkannten Spracheingabe (*ASR*) in einen Text, der Ausgabe von Informationen an den Nutzer (*GUI*) und der Steuerung des Fernsehers (*Cable Box*) zuständig. Die von der automatischen Spracherkennung erkannte Eingabe leitet der Client in einem ersten Schritt an den Server weiter. Dieser ist für die Interpretation der Spracheingabe (*Named Entity Recognizer – NER, Relation Extraction – REL*), das Abfragen von Ergebnissen (*Canonicalizer, Semantic Query Engine, Electronic Program Guide – EPG*), die Dialogsteuerung (*Belief Tracker, Knowledge Scaling, Dialog Manager, Cyc*) und die Generierung von Sprachausgaben (*Response Generation*) verantwortlich. Im Folgenden betrachten wir den Interpretationsprozess anhand der Komponenten *NER* und *REL* sowie die allgemeine Funktionsweise der Dialogsteuerung durch den *Belief Tracker*.

Interpretationsprozess

Der Server erhält zunächst den erkannten Text vom Client. Ein *Entitäten-Erkenner* versucht, in dem Text bekannte Substantive (Filmtitel, Personennamen, etc.) sowie andere Phrasen (z. B. Genre oder Zeitangaben) zu detektieren und ordnet ihnen Entitäten aus einem Entitätsmodell zu. Der Erkenner basiert auf dem maschinellen Lernverfahren der *Maximalen Entropie* (vgl. [4]). Nachdem der Erkennungsprozess abgeschlossen ist, werden die erkannten *Phrase-Entität-Paare* an die *Relationen-Extraktions*-Komponente weitergeleitet. Anhand eines hinterlegten Wissensmodells[5] generiert das System aus den Phrase-Entität-Paaren einen *beschrifteten relationalen Wurzelbaum* (vgl. Abbildung 6).

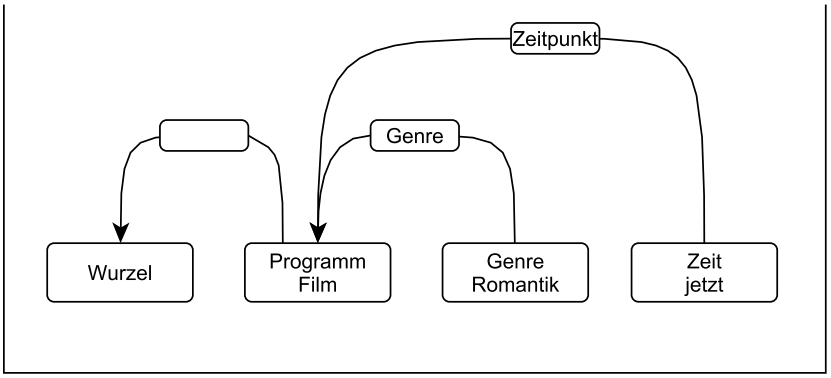


Abbildung 6: Beschrifteter relationaler Wurzelbaum im Kellerspeicher zu der erkannten Sprachäußerung: „Ich möchte jetzt einen romantischen Film sehen.“.

Diese semantische Datenstruktur ermöglicht, im Gegensatz zu den häufig verwendeten Slot-Werte-Paaren, eine detailliertere Darstellung der erkannten Sprachäußerung (vgl. [18]). Während die Erweiterung der Slotanzahl eine Implementierung neuer Regeln nach sich zieht, ist beim gewählten Ansatz die Anpassung des hinterlegten Wissensmodells notwendig. Des Weiteren kann der beschriftete Wurzelbaum in anschließenden Dialogschritten ergänzt beziehungsweise angepasst werden. Die dafür notwendigen Aktualisierungsoperationen sind *Hinzufügen*, *Löschen* und *Vereinheitlichung*.

Dialogsteuerung

Das System verwendet zur Repräsentation des Dialogzustands einen Kellerspeicher. Existiert innerhalb des Kellerspeichers kein beschrifteter Wurzelbaum, speichert das System den aktuellen beschrifteten Wurzelbaum als T_0 in diesem ab und versucht durch eine Datenbankabfrage die passenden Informationen zu finden. Existiert jedoch bereits ein beschrifteter Wurzelbaum T_0 aus einem vorherigen Dialogschritt, versucht das System eine

[5]Das Modell wurde in einer Trainingsphase erlernt.

Menge von Aktualisierungsregeln – die aus zwei *Baum-Regulär-Ausdrücke*, einer Menge von *Baum Einschränkungen* und einer *Transformationsvorschrift* bestehen – (vgl. [18]) auf den aktuellen T_1 und den alten beschrifteten Wurzelbaum T_0 anzuwenden. Das Ziel ist dabei die Erweiterung der bisher vom System interpretierten Nutzerintention. Schlägt die Anwendung der Aktualisierungsregeln fehl, geht das System von einer neuen Nutzerintention aus und speichert den aktuellen beschrifteten Wurzelbaum T_1 als oberstes Element in den Kellerspeicher. Können die Aktualisierungsregeln im folgenden Dialogschritt nicht auf T_1 und einer weiteren erkannten Benutzerintention T_2 angewendet werden, versucht das System, diese auf T_0 und T_2 anzuwenden. Bei der erfolgreichen Ausführung einer Aktualisierungsregel fand aus Sicht des Systems ein Wechsel der Nutzerintention statt. Der bearbeitete beschriftete Wurzelbaum wird anschließend in eine *SQL*-Anfrage umgewandelt und weiter verarbeitet. Ein Beispiel des vollständigen Ablaufs ist in [18] beschrieben. Mit der Erweiterung des Dialogzustands zu einem *Dialog-Annahme-Zustand* (engl. *dialog belief state*) können verschiedene Erkennungshypothesen bezüglich des automatischen Spracherkennungs- und des Interpretationsprozesses berücksichtigt werden. Der angenommene Dialogzustand besteht aus einer Wahrscheinlichkeitsverteilung über eine endliche Anzahl von Kellerspeicherzuständen, die beschriftete Wurzelbäume enthalten.
Die hier entwickelte Verhaltenssteuerung ist ein Hybrid aus Komponenten zur Dialog- und Heizungssteuerung. Mit den beschrifteten gewichteten Merkmal-Werte-Relationen kommen ebenfalls semantische Datenstrukturen aus einer festgelegten Ontologie zum Einsatz, die als Wurzelbäume interpretierbar sind. Im Gegensatz zu dem System von *Nuance Communication Inc.* verwendet die entwickelte Verhaltenssteuerung semantische Datenstrukturen ebenfalls zur Steuerung des Endgerätes und der Generierung von Sprachausgaben an den Nutzer. Die vom System getätigten Schlussfolgerungen basieren somit ausschließlich auf semantischen Datenstrukturen, während sie im Dialogsystem zur Auswahl eines TV-Programms für die Erstellung von Datenbankanfragen dienen. Des Weiteren können auszuführende Systemaktionen in die Zukunft verlegt werden.

2.2. Semantikrepräsentation mit bgMWR

Zur Verarbeitung der beschrifteten gewichteten Merkmal-Werte-Relations-Graphen in der Verhaltenssteuerung kommen verschiedene Algorithmen zum Einsatz. Für eine mathematische Beschreibung sind die Operationen *bgMWR-Hinzufügen*, *bgMWR-Löschen* und *bgMWR-Vereinheitlichung* definiert worden. Diese setzen sich aus Mengenoperationen auf Graphen zusammen. Die grundlegenden Aussagen zu den mathematischen Strukturen und Eigenschaften von Graphen sowie beschrifteten gewichteten Merkmal-Werte-Relationen dienen zur abgeschlossenen Betrachtung.

2.2.1. Strukturen und Eigenschaften

Ein *gerichteter Graph* ist ein Tupel

$$G := (V, E), \tag{2.2.1}$$

wobei die endliche Menge V alle *Knoten* und die Menge $E \subseteq V \times V$ alle *Kanten* des Graphen G beinhaltet. Eine Kante $(u, v) \in E$ wird im Folgenden kurz uv notiert. Für einen Knoten $v \in V$ ist $\Gamma_< (v) := \{u \mid uv \in E\}$ die Menge seiner direkten *Vorgänger* und $\Gamma_> (v) := \{u \mid vu \in E\}$ die Menge seiner direkten *Nachfolger*. Aus der Endlichkeit von V folgt, dass auch E endlich ist und damit sind auch $\Gamma_<$ und $\Gamma_>$ endlich. Die *Nachbarschaft* eines Knotens $v \in V$ setzt sich aus seinen Vorgängern und seinen Nachfolgern zusammen

$$\Gamma (v) := \Gamma_< (v) \cup \Gamma_> (v). \tag{2.2.2}$$

Der *Prägrad* eines Knotens $v \in V$ entspricht der Anzahl seiner Vorgänger

$$deg_< (v) := |\Gamma_< (v)|,$$

der *Postgrad* der Anzahl seiner Nachfolger

$$deg_> (v) := |\Gamma_> (v)|$$

und der *Grad* (engl. *degree*) der Größe seiner Nachbarschaft

$$deg (v) := |\Gamma (v)| = deg_< (v) + deg_> (v) - |\Gamma_< (v) \cap \Gamma_> (v)|. \tag{2.2.3}$$

Eine Folge $(v_0, \ldots, v_l)$ von Knoten in V heißt ein *Weg*, wenn für alle $i = 1, \ldots, l$ gilt, dass $v_{i-1}v_i$ eine Kante aus E ist. Die *Länge* des Weges ist l, v_0 heißt *Startknoten*, v_l *Endknoten*. Ein Weg, dessen Endknoten v_l keinen Nachfolger hat – also $deg_> (v_l) = 0$ – heißt *maximal*. Ein maximaler Weg, dessen Startknoten v_0 keinen Vorgänger hat – also $deg_< (v_0)$ heißt *durchgehend*. Für einen Weg $(v_0, \ldots, v_l)$ ist jeder Weg $(v_0, \ldots, v_i)$ mit $i = 0, \ldots, l$ ein *Präfix* und $(v_{i+1}, \ldots, v_l)$ der *entsprechende Rest*. Für zwei Wege bildet der Durchschnitt der Mengen ihrer Präfixe die *gemeinsamen Präfixe*. Zwei Wege heißen *ähnlich*, wenn ihre gemeinsamen Präfixe nicht leer sind. Die *Ähnlichkeit* zweier Wege ist die Länge l des längsten gemeinsamen Präfixes. Sind alle Knoten eines Weges paarweise verschieden – gibt es also keine Kreise – nennt man den Weg einen *Pfad*. Sind alle Wege eines Graphen Pfade, heißt der Graph *kreisfrei*. Ein Pfad der Länge 0, der nur aus einem Knoten besteht, heißt *trivialer Pfad*.

Für einen Knoten $v \in V$ bezeichnet $R(v)$ die Menge aller Pfade in G, deren Startknoten v ist. Mit $R^{>}(v)$ werden alle maximalen Pfade in G bezeichnet, deren Startknoten v ist. Dabei gilt stets $R^{>}(v) \subseteq R(v)$. Für eine Menge R von Pfaden in G bezeichnet $V(R)$ die Menge aller Knoten, die auf einem Pfad aus R liegen. Entsprechend bezeichnet $E(R)$ die Menge aller Kanten, die in Pfaden aus R auftreten. Für einen einzelnen Pfad r schreiben wir statt $V(\{r\})$ die verkürzte Variante $V(r)$ und $E(r)$ für $E(\{r\})$. Ein gerichteter Graph $G = (V, E)$ heißt *zusammenhängend*, wenn für je zwei Knoten $v, w \in V$ ein Pfad $(v, \ldots, w)$ in $G' := (V, E')$ existiert, wobei E' der symmetrische Abschluss der Relation E ist. Für einen gerichteten, zusammenhängenden, kreisfreien Graph $T = (V, E)$ heißt (T, v) ein *Wurzelbaum*, wenn $v \in V$ der Startknoten eines jeden durchgehenden Pfades in T ist. Der Knoten v heißt dann *Wurzel*. Für ein $l \in \mathbb{N}$ liegen alle Knoten der Menge $\{w \in V \mid \exists\, (v, \ldots, w) \in R(v) \text{ der Länge } l\}$ auf der selben *Ebene*. Ein Graph kann des Weiteren an den Knoten oder an den Kanten beschriftet werden. Dazu benötigt man eine Menge A mit Beschriftungen und eine Abbildung $\nu : V \to A$ bzw. eine Abbildung $\lambda : E \to A$. Für eine beschriftete Zeichenkette $s(r) = s(v_0, \ldots, v_l) = \big(\nu(v_0) \circ \ldots \circ \nu(v_l)\big) = \nu(v_0, \ldots, v_l) = \nu(r)$, $s(r) \in A^*$ des Pfades r gelten die Bedingungen des *Zeichenketten-Halbrings* (vgl. [9])

$$\mathbb{S} = (A^* \cup \{\infty\}, \wedge, \cdot, \infty, \epsilon) \qquad \text{mit } s \wedge \infty = s, \forall s \in A^*. \tag{2.2.4}$$

$A^* \cup \{\infty\}$ beschreibt die Menge aller Zeichenketten, die durch beliebige Konkatenation der Beschriftungen in A gebildet werden können. ∞ ist das neutrale Element zur Operation $\wedge$ (längste gemeinsame Präfix) und ϵ das neutrale Element zur Verkettungsoperation $\cdot$.

2.2.2. Merkmal-Werte-Relationen und Ausprägungen

Eine Merkmal-Werte-Relation *mwr* ist eine zyklenfreie Relation (vgl. [15]). Sie besteht aus den Elementen der beiden disjunkten endlichen Mengen M – Menge der *Merkmale* – und W – Menge der *atomaren Werte* – mit

$$mwr \subset M \times (M \cup W). \tag{2.2.5}$$

Existiert zu einer MWR *mwr* eine Beschriftungsfunktion $\nu : M \cup W \to A$, die jedem Merkmal und jedem atomaren Wert eine Beschriftung aus der Menge A von *semantischen Ankern*[6] zuordnet, handelt es sich um eine *beschriftete Merkmal-Werte-Relation*

$$bmwr := (mwr, \nu). \tag{2.2.6}$$

[6] Ein *semantischer Anker* ist nach [27] ein subjektiver Bezug auf eine subjektive Wirklichkeit.

Alle Elemente $\{e|\exists(m,e) \in mwr\}$ die zu einem Merkmal $m \in M$ in Relation stehen, besitzen unterschiedliche Beschriftungen.
Durch die Erweiterung der beschrifteten Merkmal-Werte-Relation *bmwr* mit einer Gewichtsfunktion $\kappa : M \cup W \to \mathbb{R}_+$, die jedem Merkmal und Wert einen Konfidenzwert aus dem *Konfidenzhalbring* $\mathbb{C} = (\{-\infty\} \cup [0,\ 1], max, \uplus, -\infty, 0)$ nach [26] zuordnet, erhalten wir eine *beschriftete gewichtete Merkmal-Werte-Relation*

$$bgmwr := (bmwr, \kappa). \tag{2.2.7}$$

In dieser Arbeit werden ausschließlich bgMWRen verwendet und durch Wurzelbäume dargestellt. Damit dies möglich ist, gehen wir davon aus, dass eine bMWR immer genau ein Merkmal m_0 besitzt, das nur auf der linken Seite der Relation vorkommt (*Wurzel*). Dies ist ohne Beschränkung der Allgemeinheit unter der Annahme möglich, dass *bmwr* eine beschriftete Merkmal-Werte-Relation ist und $\mathcal{M} \subseteq M$ die Menge derjenigen Merkmale, welche nur auf der linken Seite der Relation auftreten. Wir erweitern dafür M mit einem zusätzlichen Merkmal $m_0 \notin M$ und die Relation mit $\{m_0\} \times \mathcal{M}$.

Zu einer beschrifteten gewichteten Merkmal-Werte-Relation *bgmwr* mit der Merkmalsmenge M, der Wurzel m_0 und der Menge der atomaren Werte W beschreibt

$$G\,(bgmwr) := (T, m_0, \nu, \kappa), \tag{2.2.8}$$

mit dem gerichteten Graphen $T = (M \cup W, mwr)$ den zugehörigen *bgMWR-Graph*.

Zur besseren Unterscheidung zwischen den Elementen von M und W sind alle Knoten, die einen atomaren Wert repräsentieren, als Rechteck und alle Knoten, die ein Merkmal darstellen als Ellipse in den folgenden Abbildungen notiert (siehe beispielsweise Abbildung 7). Die Beschriftung und das Gewicht eines Knotens sind durch einen Doppelpunkt getrennt.

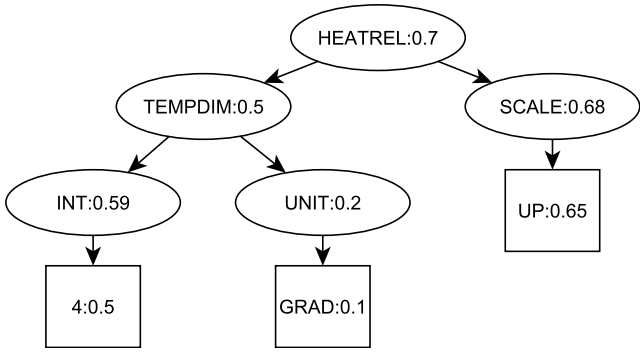


Abbildung 7: Visualisierung eines bgMWR-Graphen zu der vom Nutzer getätigten Sprachäußerung „Temperatur um 4 Grad erhöhen“.

2.2.3. Operationen auf bgMWR-Graphen

bgMWR-Hinzufügen

Im Folgenden sind r_1 und r_2 zwei durchgehende Pfade sowie q das längste gemeinsame Präfix von $\nu_1(r_1)$ und $\nu_2(r_2)$. Daraus ergibt sich, dass q eine Folge in A ist. Aufgrund der eingangs erwähnten Voraussetzung, dass eine *bmwr* und somit ebenfalls eine *bgmwr* das Merkmal m_0 besitzt, ist q mindestens der triviale Pfad aus der gemeinsamen Wurzel. Wir bezeichnen die Präfixe von r_1, r_2 als q_1, q_2 mit $\nu_1(q_1) = \nu_2(q_2) = q$ und $\tilde{r}_1$, $\tilde{r}_2$ die entsprechenden Reste von r_1 sowie r_2 sind. Des Weiteren ist e der Endknoten von q_1 und s der Startknoten von $\tilde{r}_2$. Der beschriftete gewichtete Merkmal-Werte-Relations Graph

$$(T, m_0, \nu, \kappa) =: r_1 \oplus r_2 \tag{2.2.9}$$

ist wie folgt definiert:

- $T = (V, E)$ mit
  - $V = V(r_1) \cup V(\tilde{r}_2)$
  - $E = E_1|_{V(r_1)} \cup E_2|_{V(\tilde{r}_2)} \cup (e, s)$
- $m_0 = m_{0,1}$
- ν setzt sich zusammen aus
  - $\nu_1|_{V(r_1)}$
  - $\nu_2|_{V(\tilde{r}_2)}$
- κ setzt sich zusammen aus
  - Konfidenzwerte des r_1 Rests $\kappa_1|_{V(\tilde{r}_1)}$,
  - Konfidenzwerte des r_2 Rests $\kappa_2|_{V(\tilde{r}_2)}$ und
  - neu berechnete Konfidenzwerte[7] der vom Präfix q_1 abhängigen Knotenmenge $V(q_1)$, $\kappa_\oplus : V(q_1) \to \mathbb{R}_+$ mit

    $$v_{i,1} \mapsto \kappa_1(v_{i,1}) + \kappa_2(v_{i,2}) - \kappa_1(v_{i,1}) \cdot \kappa_2(v_{i,2}),$$

    wobei $v_{i,2}$ derjenige Knoten aus q_2 ist, der an der gleichen Position wie $v_{i,1}$ in q_1 steht.

[7] In dieser Arbeit wird dafür der *Konfidenzhalbring* nach [26] für Merkmale und Werte verwendet.

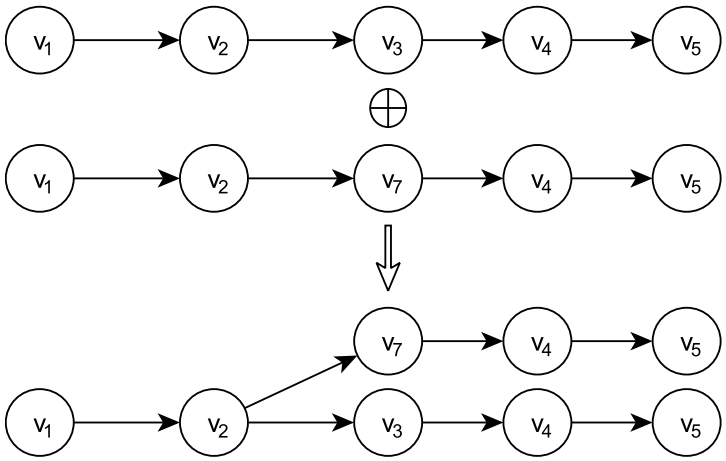


Abbildung 8: Ausführungsbeispiel der Operation bgMWR-Hinzufügen ohne Betrachtung der Konfidenzwerte.

bgMWR-Löschen

Für einen bgMWR-Graph G_1 und einen Knoten $v \in V_1 \setminus \{m_{0,1}\}$ ist der bgMWR-Graph

$$(T, m_0, \nu, \kappa) =: G_1 \ominus v \tag{2.2.10}$$

wie folgt definiert:

- $T = (V, E)$ mit
  - $V = V_1 \setminus V\left(R\left(v\right)\right)$
  - $E = E_1|_V$
- $m_0 = m_{0,1}$
- $\nu = \nu_1|_V$
- $\kappa = \kappa_1|_V$

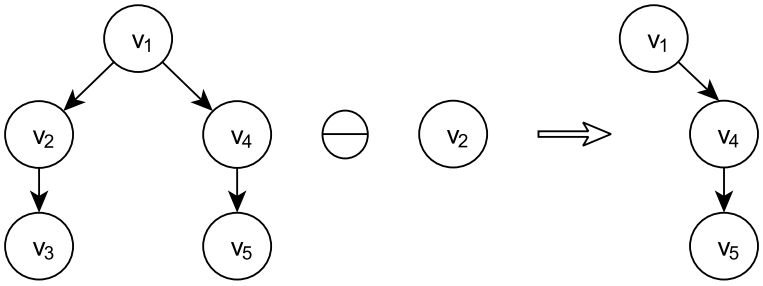


Abbildung 9: Ausführungsbeispiel der Operation bgMWR-Löschen ohne Betrachtung der Konfidenzwerte.

bgMWR-Vereinheitlichung

Für zwei bgMWR-Graphen G_1 und G_2, für die entweder $\left|\nu_1^{-1}\left(\nu_2\left(m_{0,2}\right)\right)\right| = 1$ oder $\left|\nu_2^{-1}\left(\nu_1\left(m_{0,1}\right)\right)\right| = 1$ – die Beschriftung der Wurzel des einen soll also genau einmal als Beschriftung im anderen vorkommen – gilt, entsteht der bgMWR-Graph

$$(T, m_0, \nu, \kappa) =: G_1 \uplus G_2$$

nach folgendem Ablauf:

- Setze die Hilfsmengen für bgMWR-Graphen $U := \{\}$, für Pfade mit *maximaler Ähnlichkeit* $R := \{\}$ und für *Knoten-Konfidenz-Tupel* $K := \{\}$.
- Wenn die Beschriftung der Wurzeln gleich ist $\nu_1\left(m_{0,1}\right) = \nu_2\left(m_{0,2}\right)$:
  - Setze $R_1 := R^{>}\left(m_{0,1}\right)$ und $R_2 := R^{>}\left(m_{0,2}\right)$
  - $m_0 = m_{0,1}$
- Wenn die Beschriftungen der Wurzeln ungleich ist $\nu_1\left(m_{0,1}\right) \neq \nu_2\left(m_{0,2}\right)$, findet zunächst eine Vorverarbeitung statt.
  - Ist die Beschriftung der Wurzel von G_2 genau einmal in G_1 enthalten:
    - bestimmte den Knoten v_i in G_1 mit der Beschriftung, die äquivalent zur Beschriftung der Wurzel in G_2 ist und setze $U \to U \cup \{G_1 \ominus v_i\}$
    - bestimmte $R_1 := R^{>}\left(v_i\right)$ als die Menge von maximalen Pfaden mit dem Startknoten v_i in G_1 und $R_2 := R^{>}\left(m_{0,2}\right)$ als die Menge von maximalen Pfaden mit dem Startknoten $m_{0,2}$ in G_2
    - $m_0 = m_{0,1}$
  - Ist die Beschriftung der Wurzel von G_1 genau einmal in G_2 enthalten:
    - bestimmte den Knoten v_i in G_2 mit der Beschriftung, die äquivalent zur Beschriftung der Wurzel in G_1 ist und setze $U \to U \cup \{G_2 \ominus v_i\}$
    - bestimmte $R_1 := R^{>}\left(m_{0,1}\right)$ als die Menge von maximalen Pfaden in G_1 mit der Wurzel $m_{0,1}$ als Startknoten und $R_2 := R^{>}\left(v_i\right)$ als die Menge von maximalen Pfaden in G_2, die mit dem Startknoten v_i beginnen
    - $m_0 = m_{0,2}$

- Setze für jedes $r_2 \in R_2$
  - die Menge von *Pfad-Ähnlichkeits-Tupel*

$$Q_{r_2} := \{(r, l)\} = \{(\underbrace{r_1 \in R_1 | \nu_1(r_1) \wedge \nu_2(r_2) \neq \epsilon}_{r}, \underbrace{|\nu_1(r_1) \wedge \nu_2(r_2)|}_{l})\},$$

  - ermittle den *Maximalen Ähnlichkeitswert* in der Menge Q_{r_2}

$$\hat{l} = \max_{q_{r_2} \in Q_{r_2}} l(q_{r_2}),$$

  - bestimme $\hat{Q}_{r_2}$ als die Menge von Pfaden mit maximaler Ähnlichkeit aus Q_{r_2}

$$\hat{Q}_{r_2} := \{r(q_{r_2}) | q_{r_2} \in Q_{r_2} \wedge l(q_{r_2}) = \hat{l}\} \text{ und}$$

  füge schließlich $r \oplus r_2$ für genau ein beliebiges $r \in \hat{Q}_{r_2}$ in $U \rightarrow U \cup \{r \oplus r_2\}$ sowie r in $R \rightarrow R \cup \{r\}$ ein. Füge des Weiteren alle Knoten aus r, die zum gemeinsamen Präfix von $\nu_1(r)$ und $\nu_2(r_2)$ gehören, zusammen mit ihrer neuen Konfidenz als Tupel (v, c) in $K \subseteq V \times \mathbb{R}_+$ ein.

Nun können die einzelnen Bestandteile des bgMWR-Graphen bestimmt werden.

- $T = (V, E)$ mit
  - $V = \bigcup_{u \in U} V_u \cup V(R_1 \setminus R)$ und
  - $E = \bigcup_{u \in U} E_u \cup E(R_1 \setminus R) \cup E_1|_V \cup E_2|_V$.
- m_0 wurde bereits festgelegt.
- ν setzt sich zusammen aus
  - $\nu_1|_{V_1}$ und
  - $\nu_2|_{V_2}$.
- κ setzt sich zusammen aus
  - $\kappa_1|_{V \setminus K'}$ mit $K' := \{v \in V \mid \exists (v, c) \in K\}$,
  - $\kappa_2|_V$ und
  - $\kappa_{\uplus} : K' \rightarrow \mathbb{R}_+$ mit $v \mapsto c$ wobei $(v, c) \in K$ gilt.

Abbildung 10 zeigt das Ergebnis der Operation auf zwei bgMWR-Graphen als Beispiel.

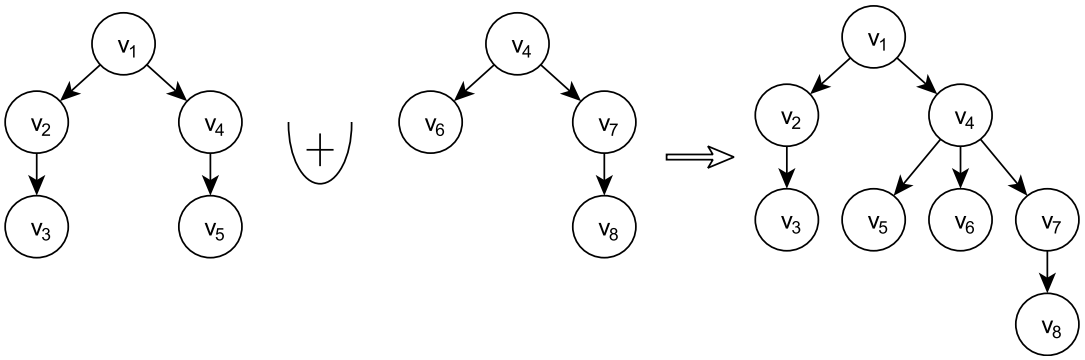


Abbildung 10: Ergebnis der Operation gMWR-Vereinheitlichung unter der Verwendung von zwei bgMWR-Graphen ohne Betrachtung der Konfidenzwerte.

Das Ergebnis der Operationen sind jeweils ein neuer bgMWR-Graph zu einer neuen bgMWR $G_{Neu}(bgmwr_{Neu})$. In [15] besitzt eine MWR per Definition die Eigenschaft *linkstotal*. Diese wurde in der vorliegenden Arbeit jedoch nicht mit in die Definition aufgenommen. Der Ausschluss erfolgte, damit die entwickelte Verhaltenssteuerung Fragestellungen akzeptieren und *Erwartungshaltungshaltungen* (ebenfalls bgMWR-Graphen) erzeugen kann. Die Blätter der bgMWR-Graphen können dabei ein Merkmal aus der Menge M sein und besitzen keinen weiteren Nachfolger. Anhand der Konstruktionsvorschriften ist das Ergebnis zwar offensichtlich, wegen der fehlenden Linkstotalität jedoch nicht mehr eindeutig. Das bedeutet: die Operation bgMWR-Vereinheitlichung mit $G_1 \uplus G_2$ liefert ein eindeutiges Ergebnis G_3. Wenn jedoch zwei bgMWRen $bgmwr_1$, $bgmwr_2$ existieren, für die $G_1 = G(bgmwr_1)$, $G_2 = G(bgmwr_2)$ gilt, kann wegen fehlender Linkstotalität nicht mehr eindeutig von G_3 auf $bgmwr_3$ und damit $G_3 = G(bgmwr_3)$ geschlossen werden. Die Blätter des Graphen sind nämlich nicht mehr automatisch Elemente aus der Menge atomarer Werte W, weshalb die Mengen M und W für $bgmwr_3$ nicht mehr eindeutig bestimmt sind. Es existieren also verschiedene $bgmwr_i$ mit $i \in \mathbb{N}$, für die $G_3 = G(bgmwr_i)$ gilt. Abbildung 11 veranschaulicht die entstehende Problematik für die Implementierung. Der konkrete bgMWR-Graph in der linken Abbildungshälfte besitzt ausschließlich atomare Werte als Blätter, wenn die Eigenschaft linkstotal als Voraussetzung für Merkmal-Werte-Relationen gilt. Der semantische Anker *TEMPIS* gehört jedoch eigentlich zur Merkmalmenge. Der bgMWR-Graph in der rechten Abbildungshälfte, besitzt den semantischen Anker *TEMPIS* nach der hier aufgestellten Definition als Knoten, welcher der Merkmalmenge angehört und keine weiteren Nachfolger besitzt. Der implementierte Algorithmus zum Übersetzen der *bgMWR-Zeichenkette* in einen bgMWR-Graph (vgl. 4.1.1) ist jedoch nicht in der Lage, zwischen atomaren Werten und Merkmalen bei Knoten ohne Nachfolger zu unterscheiden. Dementsprechend teilt dieser alle semantischen Anker einer bgMWR-Zeichenkette ausschließlich in die Kategorien *Blatt* und *innerer Knoten* ein.

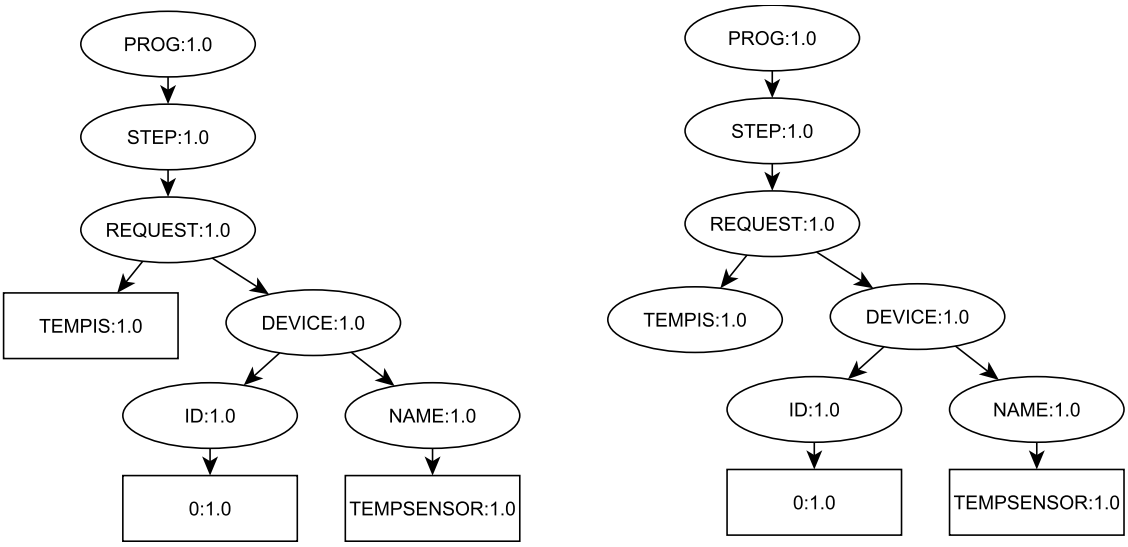


Abbildung 11: Darstellung eines konkreten bgMWR-Graphen zur Sprachäußerung: „Wie hoch ist die Raumtemperatur“, wenn die Voraussetzung linkstotal gilt (links) und wenn sie nicht gilt (rechts). Die Realisierung basiert nicht auf dem verwendeten Sprachmodell, sondern soll lediglich die Problematik darstellen.

2.3. Interpretation & Artikulation

Das Ziel der automatischen Interpretation ist die Übersetzung einer Zeichenfolge (syntaktische Ebene) des Kommunikationspartners in eine beschriftete gewichtete Merkmal-Werte-Relation (semantische Ebene). Die Artikulation beschäftigt sich hingegen mit der Generierung einer Zeichenfolgen zu der gegebenen beschrifteten gewichteten Merkmal-Werte-Relation. Eine durch den Interpretationsprozess ermittelte beschriftete gewichtete Merkmal-Werte-Relation hat eine bestimmte Auswirkung $\mathcal{C}_{Sys}$ (engl. *Consequences*) auf das System. Folgt man den Ansichten in [11], hängt diese

$$\mathcal{C}_{Sys} = f_{Interpretation}(\mathcal{B}_{Obj},\ \mathcal{A}_{Sys},\ \ldots). \tag{2.3.1}$$

von dem interpretierten Verhalten des Kommunikationspartners $\mathcal{B}_{Obj}$ (engl. *Behavior*) sowie der Vorbedingungen $\mathcal{A}_{Sys}$ (engl. *Antecedents*) des Systems und weiteren nicht näher spezifizierten Faktoren (...) ab. Die Vorbedingung setzt sich in dieser Arbeit aus den bereits ausgeführten Aktionen des Systems in der Vergangenheit und den verwendeten Modellen (ab der syntaktischen Ebene) zusammen[8]. Anhand dieser Informationen entscheidet sich das System anschließend für eine Aktion und der damit einhergehenden Erwartungshaltung an den Kommunikationspartner. Das Verhalten $\mathcal{B}_{Obj}$ des Kommunikationspartners ergibt sich durch dem dazugehörigen Objektziel $\mathcal{C}_{Obj}$, der Vorgeschichte

[8]Eine allgemeinere Betrachtung schließt ebenfalls die verwendeten Modelle im Intelligenten Signalverarbeitungssystem in die Vorbedingungen mit ein.

$\mathcal{A}_{Obj}$ des Objekts und nicht näher spezifizierten Faktoren:

$$\mathcal{B}_{Obj} = f_{Artikulation}(\mathcal{C}_{Obj},\ \mathcal{A}_{Obj},\ \ldots). \tag{2.3.2}$$

$\mathcal{A}_{Obj}$ ist von der Spezifikation des Kommunikationspartners abhängig und kann sich durchaus in der Struktur gegenüber $\mathcal{A}_{Sys}$ unterscheiden. Die für das System sich ergebende Konsequenz $\mathcal{C}_{Sys}$ ist in der Regel nicht kompatibel mit dem an das Verhalten $\mathcal{B}_{Obj}$ gebundenen Objektziel $\mathcal{C}_{Obj}$ des Kommunikationspartners. Dementsprechend ist bei einer Verhaltenssteuerung, dessen Systemverhalten von einem semantischen Modell nach [27] abhängt, mit Missverständnissen während einer Mensch-Maschine-Interaktion zu rechnen. Das kann zum Beispiel die Ermittlung einer bgMWR im äußeren Perzeptionsbereich sein, die nicht mit der Vorgeschichte des Systems übereinstimmt oder die falsche Auswahl einer auszuführenden Systemaktion. Zur Aufdeckung von Missverständnissen ist die zielgerichtete Rückfrage bezüglich der auszuführenden Systemaktion (Aktion, die eine Parameteränderung einer inneren physischen Komponente nach sich zieht) an den Kommunikationspartner durchzuführen.

2.3.1. Interpretation - äußerer Kreis

Der äußere Perzeptionsbereich (siehe Abbildung 2) dient zur Ermittlung einer beschrifteten gewichteten Merkmal-Werte-Relation aus der Sprachäußerung des Benutzers $\mathcal{B}_{Usr}$ und den Vorbedingungen des Systems $\mathcal{A}_{Sys}$. Zunächst erfolgt die Analyse des akustischen Signals durch ein intelligentes Signalverarbeitungssystem nach [9]. Diese Analyse ist jedoch nicht Bestandteil der Arbeit und wird als gegeben vorausgesetzt. Als Ergebnis erhält die Interpretationskomponente einen Worthypothesegraph $\mathcal{T}_W$ (endlicher Automat). Alle durchgehenden Pfade in $\mathcal{T}_W$ entsprechen möglichen Wortfolgen, die das intelligente Signalverarbeitungssystem dem akustischen Signal zuordnet. Die Zuordnung einer beschrifteten gewichteten Merkmal-Werte-Relation zu den Wortfolgen des Worthypothesegraphen erfolgt durch die Komposition mit dem Äußerungs-Bedeutungs-Transduktor $\mathcal{T}_{UM}$:

$$\mathcal{T}_{MWR} := \mathcal{T}_W \circ \mathcal{T}_{UM}. \tag{2.3.3}$$

Dieser basiert auf einem bedeutungsorientierten Sprachmodell (siehe Abschnitt 3.1) und übersetzt das Erkennungsergebnis $\mathcal{T}_W$ in einen semantischen Hypothesegraphen, der durch den endlichen Transduktor $\mathcal{T}_{MWR}$ repräsentiert wird (vgl. [12]). Dazu muss der Äußerungs-Bedeutungs-Transduktor $\mathcal{T}_{UM}$ mindestens einen durchgehenden Pfad aus $\mathcal{T}_W$ eingabeseitig akzeptieren.

Die am Lehrstuhl Kommunikationstechnik verwendeten Algorithmen zur Sprach-Semantik-Übersetzung basieren auf endlichen Transduktoren. Diese sind jedoch nicht in der Lage, eine direkte Übersetzung der erkannten Zeichenfolge in eine beschriftete gewichtete Merkmal-Werte-Relation durchzuführen. In [28] ist daher zunächst die Übersetzung der erkannten Zeichenfolge in eine *MWR-Zeichenkette*, welche die Merkmal-Werte-Relation repräsentiert, als Zwischenschritt vorgeschlagen worden. Der semantische Hypothesegraph beinhaltet somit die zu den akzeptierten Pfaden dazugehörigen Merkmal-Werte-Relationen als *MWR-Zeichenfolgen*.
bgMWR-Zeichenketten besitzen die selbe Struktur wie *bgMWR-Zeichenketten*. Die Algorithmen zur Übersetzung müssen somit nicht angepasst werden.

bgMWR-Zeichenketten

In dieser Arbeit erhält die Verhaltenssteuerung von dem äußeren Perzeptionsbereich genau eine bgMWR-Zeichenkette aus dem Semantikhypothesegraphen $\mathcal{T}_{MWR}$, zu der erkannten Zeichenfolge. Diese liegt in der erweiterten Backus-Naur-Form (eBNF) nach [28] vor. Die Syntax der bgMWR-Zeichenketten entspricht:

FVR	= “FVR“ state
state	= “[“ states statedef states “]“
states	= [[states] state]
statedef	= label [“:“ weight]
label	:= string
weight	:= numeric | “NaN“

Alle in Anführungszeichen gesetzten Symbole gehören zu einer Menge von Terminalsymbolen X. “FVR“ ist das Startsymbol jeder bgMWR-Zeichenfolge. Eckige Klammern ohne Anführungszeichen bilden eine Option in der eBNF. Ein *string* kann durch eine Folge von Buchstaben, Ziffern oder Sonderzeichen ersetzt werden. In unserem Fall sind es die Beschriftungen von semantischen Ankern aus der Menge A. *numeric* ist mit dem zum semantischen Anker a dazugehörigen Konfidenzwert zu ersetzen. Besitzt ein semantischer Anker a keinen Konfidenzwert, ist das Gewicht mit NaN (not a number) gekennzeichnet. Anhand der Klammerstruktur kann die bgMWR-Zeichenfolge als beschrifteter gewichteter Wurzelbaum interpretiert werden. Die softwaretechnische Übersetzung einer bgMWR-Zeichenkette, die in der erweiterten Backus-Naur-Form vorliegt, in die Graphenstruktur $G(bgmwr)$ ist im Abschnitt 4.1.1 beschrieben.

2.3.2. Artikulation - äußerer Kreis

Damit das Sprachdialogsystem dem Nutzer eine Information sprachlich mitteilen kann, übersetzt es einen in der Verhaltenssteuerung erzeugten Graphen $G(bgmwr)$ zunächst in eine Wortfolge. Diese wird anschließend vom intelligenten Signalverarbeitungssystem weiterverarbeitet. Der Übersetzungsprozess ist Aufgabe des äußeren Aktionsbereiches und umfasst die drei Schritte (vgl. [13]):

1. Erstellung eines Transduktors $\mathcal{T}_{MWP}$, der alle durch Permutation generierten gültigen bgMWR-Zeichenketten zu $G(bgmwr)$ beinhaltet.
2. Bildung eines Bedeutungs-Äußerungs-Transduktors $\mathcal{T}_{MU}$ mit Sprachzeichenfolgen zu den erzeugten bgMWR-Zeichenketten in $\mathcal{T}_{MWP}$.
3. Optionale gewichtete Bewertung der generierten Sprachzeichenfolgen (z. B. mit N-Gram-Sprachmodell) sowie Auswahl einer Sprachzeichenfolge.

Die Erstellung des Transduktors $\mathcal{T}_{MWP}$ erfolgt mit einer *Linearisierung* des Graphen $G(bgmwr)$. Dabei erzeugt das System zunächst durch einen rekursiven Permutationsalgorithmus alle möglichen *Merkmal-Werte-Permutationen* in Form von bgMWR-Zeichenketten. Generierte bgMWR-Zeichenketten, die nicht der *bgmwr* entsprechen, sind nicht gültig und zu verwerfen. Alle gültigen bgMWR-Zeichenketten werden anschließend als Ausgabesymbolfolgen zur Erstellung des Transduktors $\mathcal{T}_{MWP}$ verwendet. Der Bedeutungs-Äußerungs-Transduktor $\mathcal{T}_{MU}$ akzeptiert verschiedene bgMWR-Zeichenketten und generiert Symbolfolgen der Zielsprache. Im einfachsten Fall ist eine Invertierung der Ein- und Ausgabezeichen des Äußerungs-Bedeutungs-Transduktors $\mathcal{T}_{UM}^{-1}$ möglich.
In der vorliegenden Arbeit verwendet das Sprachdialogsystem jedoch Bedeutungsstrukturen, die nicht im Äußerungs-Bedeutungs-Sprachmodell enthalten sind. Dies betrifft vor allem Fragestellungen an den Nutzer zu einem bestimmten Sachverhalt. Es erfolgt somit eine separate Modellierung des Bedeutungs-Äußerungs-Transduktors $\mathcal{T}_{MU}$ anhand eines Bedeutungs-Äußerungs-Sprachmodells. Die Komposition der beiden Transduktoren ergibt

$$\mathcal{T}_{AW} = \mathcal{T}_{MWP} \circ \mathcal{T}_{MU}. \tag{2.3.4}$$

$\mathcal{T}_{AW}$ ist ein Automat, der alle Wortfolgen zu den akzeptierten gültigen bgMWR-Zeichenketten generiert. Die Auswahl einer bestimmten generierten Wortfolge kann nach [12] durch die Bewertung mit einem statistischen Sprachmodell $\mathcal{T}_{SM}$ erfolgen. Enthält der Automat $\mathcal{T}_{AW}$ keine oder nur schlecht bewertete Wortfolgen, teilt die Artikulations-Komponente dies der Verhaltenssteuerung mit. Der bgMWR-Graph wird angepasst und anschließend erneut an die äußere Artikulations-Komponente gesendet.

2.3.3. Interpretation - innerer Kreis

Die im inneren Perzeptionsbereich erzeugten Zeichenfolgen sind im Gegensatz zum äußeren kognitiven Kreis nicht vom Nutzer abhängig, sondern von Spezifikationen der inneren Sensoren und Aktuatoren. Erhält das System ausschließlich den Wert eines Sensors oder Aktuators als Zeichenfolge, benötigt es Zusatzinformationen, um einen sinnvollen Bezug zu semantischen Ankern herzustellen. Das kann zum Beispiel der Name des Sensors bzw. Aktuators sein. Die Interpretationskomponente vergleicht anschließend die angereicherte Zeichenfolge mit allen Zeichenfolgen, die in einem *Befehlssequenz-Bedeutungs-Modell* (siehe Abschnitt 3) hinterlegt sind. Stimmt die angereicherte Zeichenfolge mit einer hinterlegten überein, wählt das System die dazugehörige bgMWR-Zeichenkette aus und setzt den konkreten Wert für den oder die dafür vorgesehenen Platzhalter ein. Analog zur Interpretation des äußeren Kreises erfolgt anschließend eine Übersetzung der bgMWR-Zeichenkette in einen bgMWR-Graph. Der Vorgang ist in Abbildung 12 am Beispiel des Raumtemperatursensors illustriert.

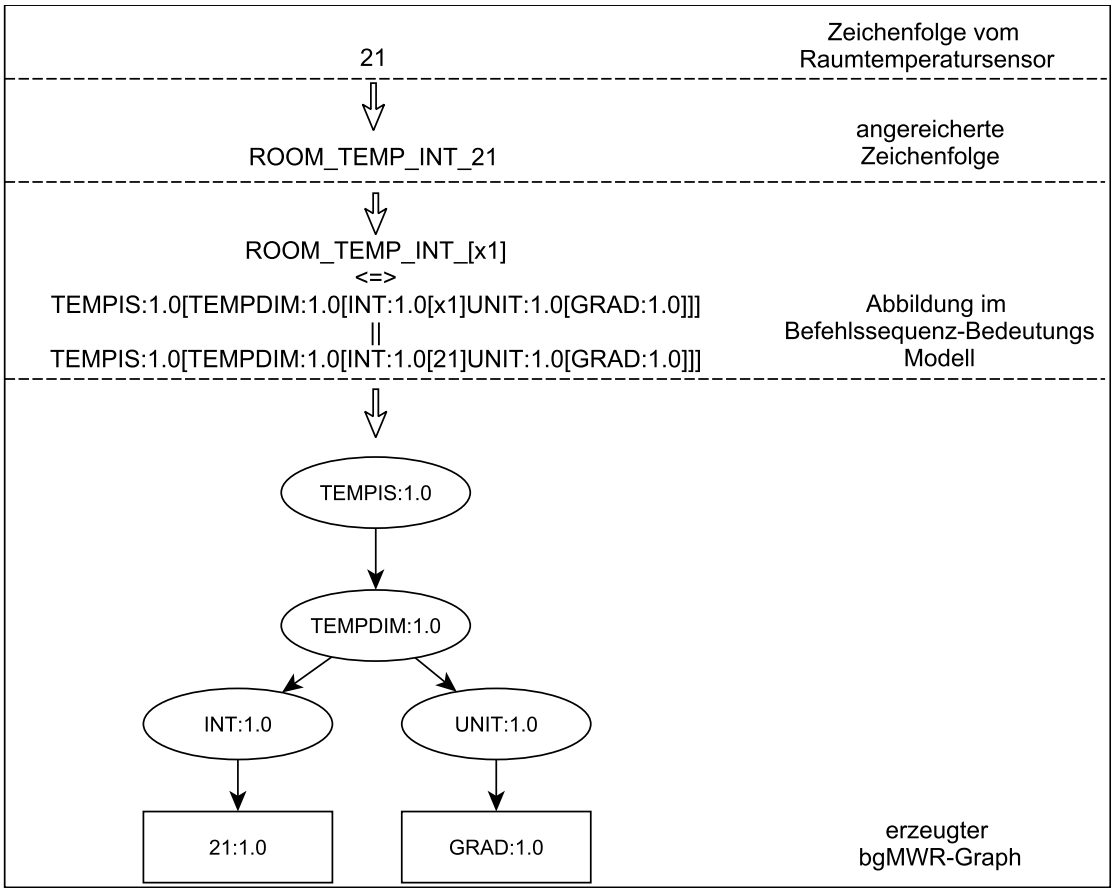


Abbildung 12: Übersetzung einer Zeichenfolge des Raumtemperatursensors in den bgMWR-Graph $G(bgmwr)$. In der vorliegenden Arbeit ist der Konfidenzwert jedes Knoten innerhalb des Graphen 1.

Die Kommunikation mit Sensoren beziehungsweise Aktuatoren ist in der Realität häufig stark eingegrenzt und beschränkt sich auf die Abfrage oder das Setzen aller Werte. In der praktischen Umsetzung kann es bei einer Sensorabfrage beispielsweise zur Rückgabe aller Sensorparameter kommen. Erwartet das System jedoch nur einen bestimmten Wert, muss die Verhaltenssteuerung diesen aus einem übergenerierten bgMWR-Graph herausfiltern und mit der Erwartung – liegt ebenfalls als bgMWR-Graph vor – vereinheitlichen. Die im *Befehlssequenz-Bedeutungs-Modell* zur Konstruktion von bgMWR-Graphen verwendeten semantischen Anker müssen ebenfalls von der Verhaltenssteuerung interpretierbar und durch den äußeren Kreis artikulierbar sein. In der Praxis kann das kommunikative Verhalten einer inneren physischen Komponente passiv oder aktiv modelliert werden. Während bei der aktiven Modellierung jede innere physische Komponente ihren Wert in einem vorgegebenen Zeitintervall an die Verhaltenssteuerung sendet[9], stellt die Verhaltenssteuerung bei der passiven Modellierung zunächst eine Anfrage an die konkrete physische Komponente und erwartet daraufhin eine entsprechende Antwort.

2.3.4. Artikulation - innerer Kreis

Die Informationsübermittlung von der Verhaltenssteuerung zu den inneren physischen Komponenten kann ebenfalls als Artikulation zwischen dem Assistenzsystem und der Außenwelt angesehen werden. In dem entwickelten Prototypen gehen wir von sehr primitiven Sensoren und Aktuatoren aus, die ausschließlich Zeichenfolgen interpretieren können, um ihren aktuellen Wert zurückzugeben oder zu setzen. Der innere Aktionsbereich dient somit zur Übersetzung eines bgMWR-Graphen in eine Befehlssequenz, die anschließend an alle inneren physischen Komponenten gesendet wird. Die Sequenz besteht in der vorliegenden Arbeit aus den Parametern *Hardwarename*, *Aufgabe* und *Wert*. Gleichzeitig erzeugt der Aktionsbereich einen bgMWR-Graph, der die Erwartungshaltung des Systems an die entsprechende physische Komponente wiedergibt. Dieser bgMWR-Graph wird an die Schnittstelle von der Verhaltenssteuerung und dem inneren Perzeptionsbereich gesendet. Nach dem Eingang einer Information, übermittelt von den physischen Komponenten, erfolgt ein Abgleich zwischen der Erwartungshaltung und der neuen Information. Zur Erzeugung der Befehlssequenz verwendet das System ein *Bedeutungs-Befehlssequenz-Modell*, wodurch eine bgMWR-Graph anhand der semantischen Anker einer entsprechenden auszuführenden Aktion zugeordnet wird. Ein beispielhafter Ablauf des Artikulations- und Interpretationsvorgangs vom inneren Kreis ist in den Abbildungen 13 bis 15 durch eine Nutzeranfrage bezüglich der aktuellen Raumtemperatur dargestellt.

[9] Dieses Vorgehen spart Zeit bei der Abfrage von Werten durch den Nutzer und ermöglicht die Umsetzung eines *proaktiven Assistenzsystems* (vgl. [7]).

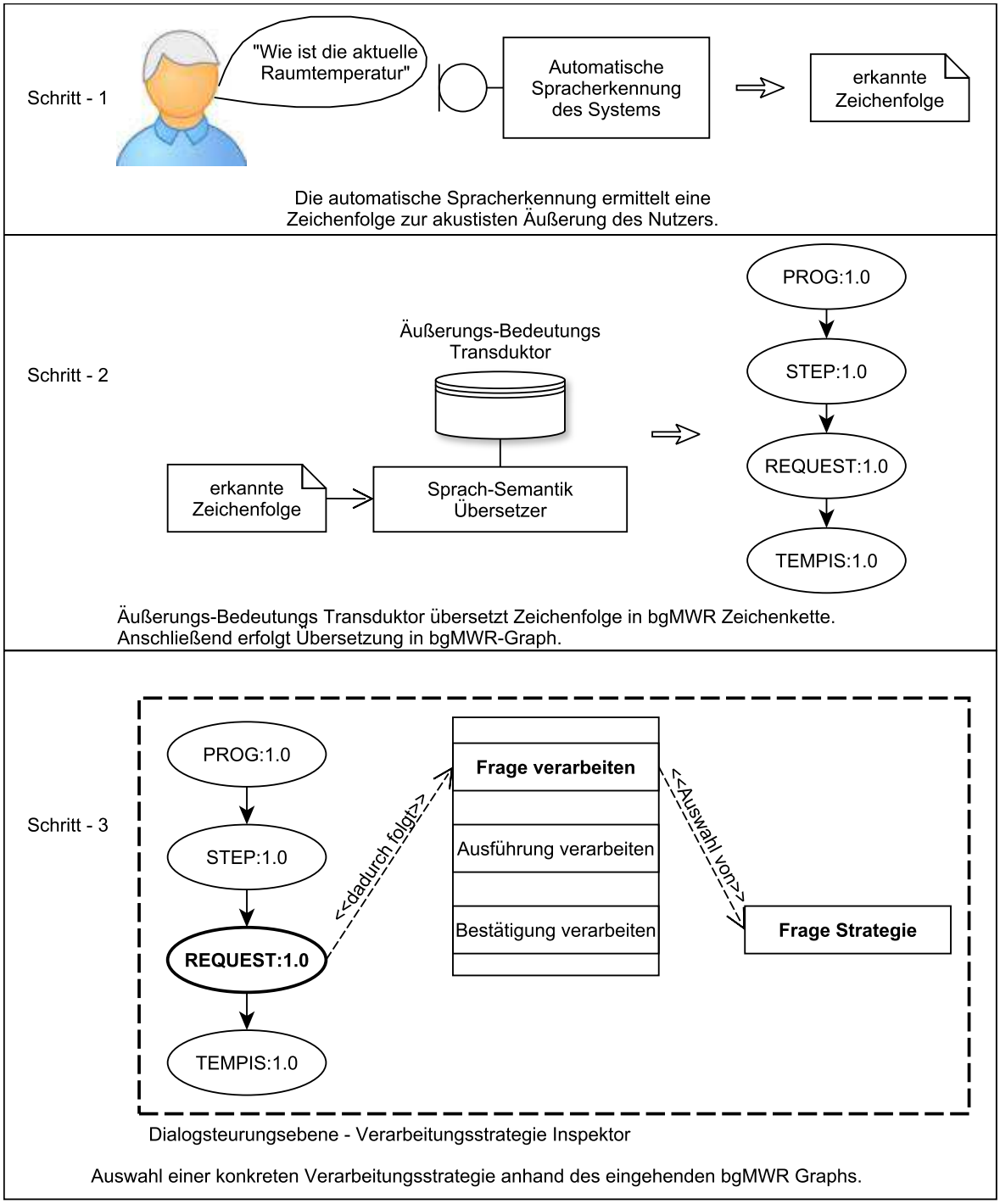


Abbildung 13: Teil 1 der hintereinander auszuführenden Prozesse zur Verarbeitung einer vom Nutzer gestellten Frage. Schritt 1: Spracherkennung und Ausgabe einer zur Äußerung gehörenden Zeichenfolge. Schritt 2: Überführung der Zeichenfolge in bgMWR-Graph. Schritt 3: Selektion der zur Bedeutung gehörenden Verarbeitungsstrategie (siehe Abschnitt 4.1.2).

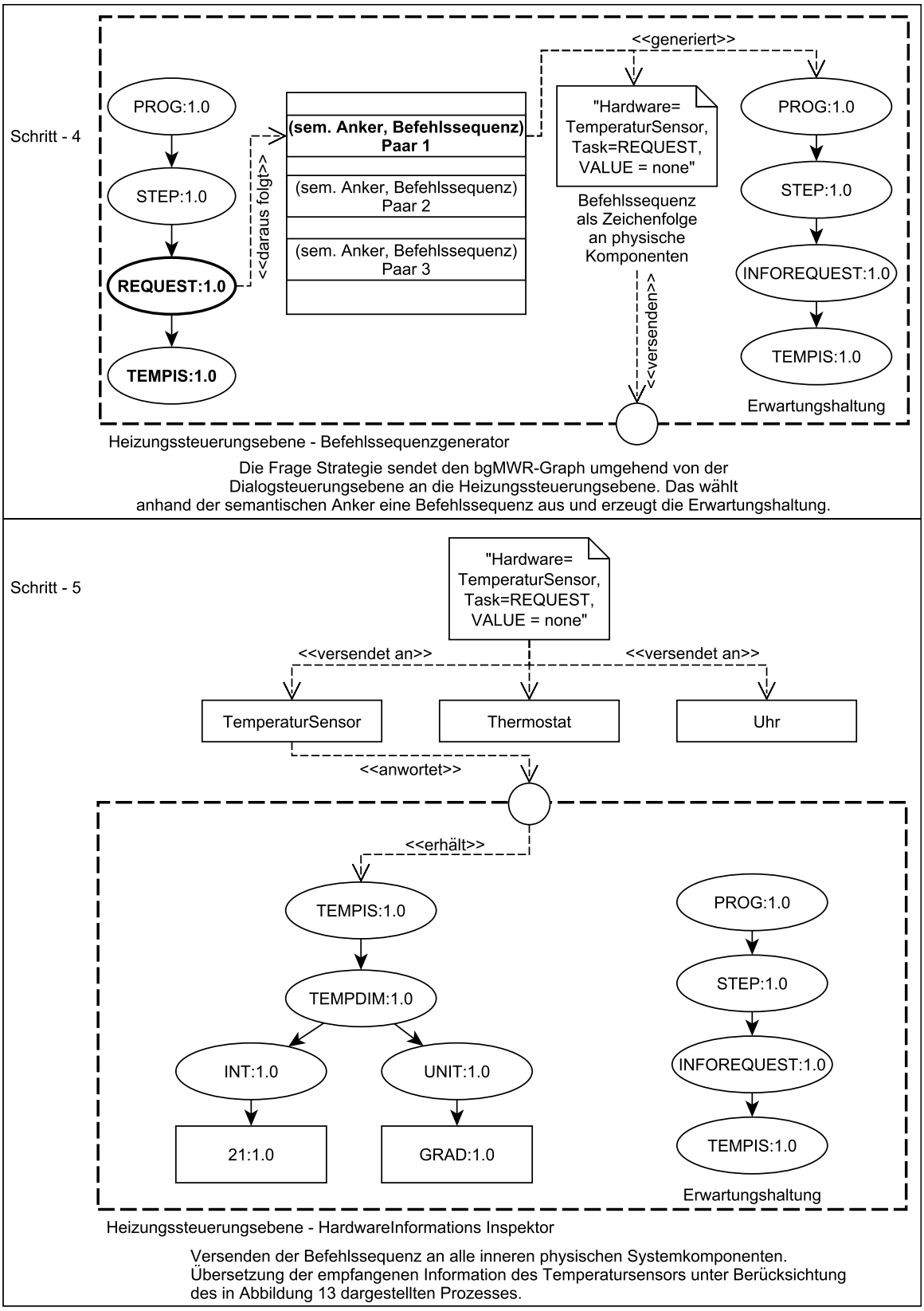


Abbildung 14: Teil 2 der hintereinander auszuführenden Prozesse zur Verarbeitung einer vom Nutzer gestellten Frage. Schritt 4: Erzeugung der Befehlssequenz als Zeichenfolge und der Erwartungshaltung als bgMWR-Graph. Schritt 5: Versenden der Befehlssequenz.

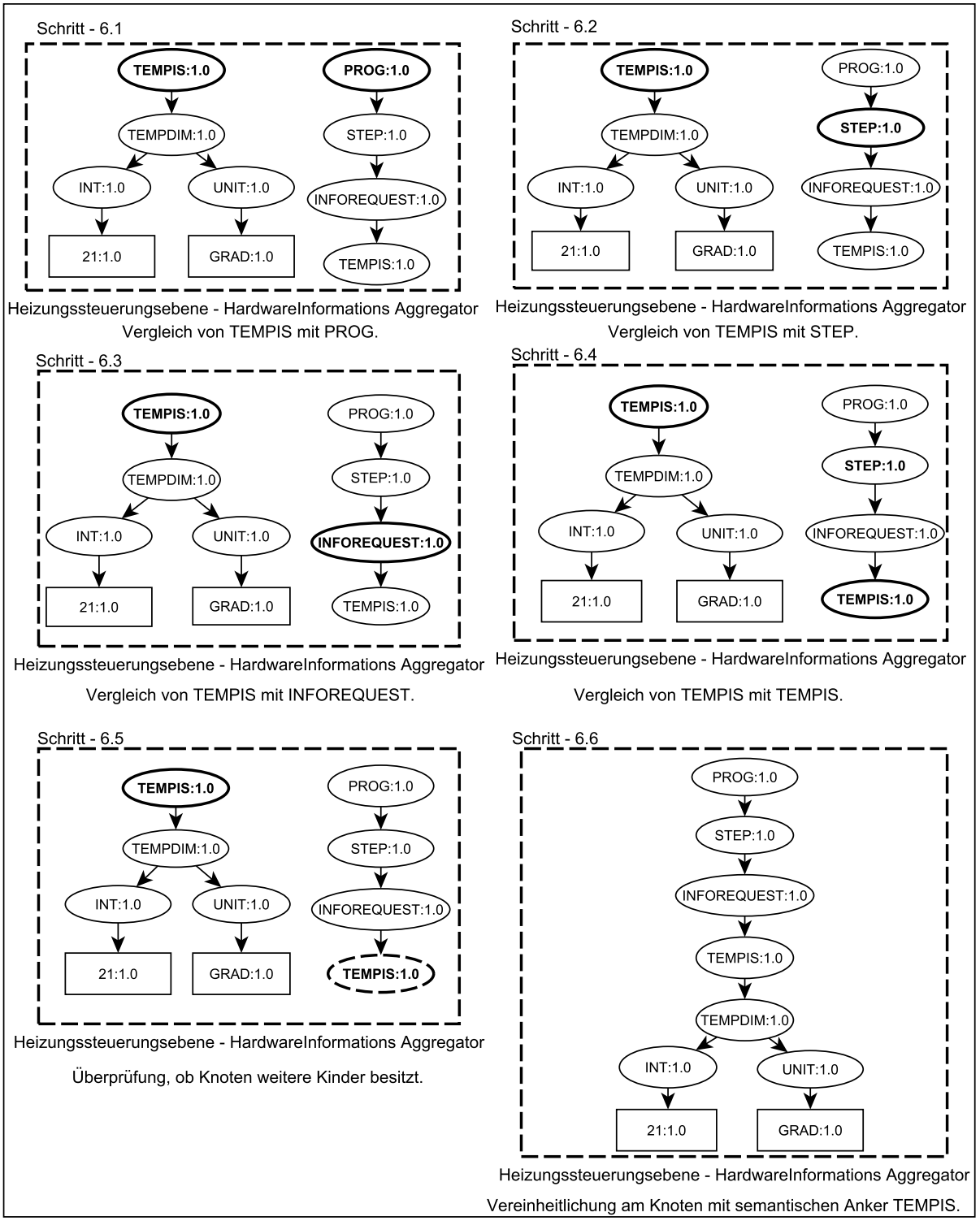


Abbildung 15: Teil 3 der hintereinander auszuführenden Prozesse zur Verarbeitung einer vom Nutzer gestellten Frage. Schritt 6: Vereinheitlichung von Erwartungshaltung (rechter bgMWR-Graph) und interpretierter Antwort der physischen Komponente (linker bgMWR-Graph).

3. Systemmodelle & Informationsstatus

3.1. Sprachmodell

Die semantische Dekodierung des Sprachsignals erfolgt mit dem in Abschnitt 2.3.1 erläuterten Äußerungs-Bedeutungs-Transduktor $\mathcal{T}_{UM}$. Dieser basiert auf einem bedeutungsorientierten Sprachmodell nach [27], das alle modellierten Spracheingaben und die dazu assoziierenden beschrifteten Merkmal-Werte-Relationen repräsentiert. Eine beschriftete Merkmal-Werte-Relation besteht aus dem kartesischen Produkt der zwei disjunkten Mengen $M \cup W$ unter den in Abschnitt 2.2.2 genannten Bedingungen. Durch die zugeordneten semantischen Anker repräsentiert sie eine *Teilbedeutung* M' für das System[10]. Zur Verwendung des Ansatzes liegt die Annahme zugrunde, dass jede natürliche Sprache endlich ist (vgl. [28]). Die sprachlichen Realisierungen $\mathcal{U}(D)$, bezüglich der Domäne D, können nach [12] durch eine reguläre Grammatik beschrieben werden. Bei der Modellierung eines Sprachmodells lässt sich die Grammatik in drei hierarchischen Ebenen unterteilen. Ein Nutzer kann durchaus unterschiedliches Verhalten $\mathcal{B}_{Usr}$ bezüglich einer Teilbedeutung M' zeigen. Die Menge aller vom automatischen Spracherkenner zu erkennenden Sprachäußerungen mit einer identischen Teilbedeutung $\mathcal{U}(M')$, bezeichnen wir nach [28] als *Elementargrammatik*. Eine Teilbedeutung M' ändert sich jedoch bereits durch die Ersetzung eines semantischen Ankers, der einen atomaren Wert repräsentiert. Es ist dementsprechend sinnvoll, Platzhalter an verschiedenen Stellen einzufügen, die zur Laufzeit durch einen konkreten semantischen Anker ersetzt werden können (vgl. [28]). Zur Modellierung von Grammatiken mit Platzhaltern ist jedoch *domänenspezifisches* sowie *linguistisches Wissen* notwendig (vgl. [12]). Mit der Substituierung eines semantischen Ankers durch einen Platzhalter ergibt sich ein semantisches Schema S. Eine Sprachäußerung u, die nach dem Ersetzen der Platzhalter zum semantischen Schema S_i passt, gehört zur Menge von Sprachäußerungen $\mathcal{U}(S_i)$, welche durch eine *mikrolokale Grammatik* $\mathcal{G}(S_i)$ definiert ist. Mit der Vereinigung aller mikrolokalen Grammatiken

$$\mathcal{G}(D) = \bigoplus_i \mathcal{G}(S_i) \tag{3.1.1}$$

erhalten wir die *lokale Grammatik* $\mathcal{G}(D)$, die den Weltausschnitt für unser System mit der Menge aller semantischen Anker A beschreibt. Für die Grammatiken kann somit nach [12] die mathematische Beziehung $\mathcal{U}(D) \supseteq \mathcal{U}(S) \supseteq \mathcal{U}(M')$ angegeben werden.

[10] Die Gesamtbedeutung aus Sicht des Systems ergibt sich erst durch das Einbeziehen den Vorbedingugen des Systems, der interpretierten Teilbedeutung des Nutzers und der internen Entscheidungslogik.

In der vorliegenden Arbeit beschränkt sich das *Äußerungs-Bedeutungs*-Sprachmodell[11] zur Steuerung des Heizungssystems auf die Systemhandlungen: Anfrage eines Sensor- oder Aktuatorwertes bezüglich der inneren physischen Komponenten Raumtemperatursensor, Thermostatsensor sowie der internen Uhr, Setzen des Thermostatwertes zu einem bestimmten Zeitpunkt und Bestätigen beziehungsweise Ablehnen eines gegebenen nicht näher spezifizierten Sachverhalts. Diese Systemhandlungen unterteilen sich in die drei Hauptschemata:

EXE	Nutzer möchte Systemaktion ausführen,
REQUEST	Nutzer fordert Information an und
CONFIRM	Nutzer trifft Aussage bezüglich Bestätigungsanfrage.

Des Weiteren existieren Teilschemata, wie beispielsweise

HEATABS	Absolute Temperaturangabe,
HEATREL	Relative Temperaturangabe oder
DATETIME	Zeit- und/oder Datumsangabe.

Vorerst ist der auf dem Sprachmodell aufbauende Äußerungs-Bedeutungs-Transduktor nur in der Lage, atomare Teilbedeutungen zu erkennen und die entsprechende bgMWR-Zeichenkette an die Verhaltenssteuerung weiterzuleiten. Zur Auswahl der Verarbeitungsstrategie (dargestellt in Abbildung 13, näher erläutert in Abschnitt 4.1.2), gehört jeder generierte bgMWR-Graph mit der obigen Einschränkung zu genau einem Hauptschema und besitzt somit den dazugehörigen semantischen Anker *EXE*, *REQUEST* oder *CONFIRM*. Neben dem Sprachmodell zur Interpretation einer sprachlichen Äußerung existiert ein weiteres zur Erzeugung der Systemantwort bezüglich eines gegebenen semantischen Sachverhaltes. Dieses Sprachmodell umfasst eine Teilmenge von semantischen Ankern, die ebenfalls zur Interpretation vorkommen und einer weiteren Menge von semantischen Ankern, die zur Artikulation im äußeren Aktionsbereich notwendig sind. Das sind in der entwickelten Verhaltenssteuerung zum Beispiel

INFOEXE	Ausführung einer Systemaktion,
INFOREQUEST	Antwort auf Nutzerfrage,
INFOCONFIRM	erforderliche Bestätigung des Nutzers und
ASKUSER	Rückfrage an den Nutzer.

Die Verarbeitung eingehender bgMWR-Zeichenketten, die dem Sprachmodell zur Interpretation entsprechen und die Generierung von Zeichenketten, welche eine Systemantwort als semantische Datenstruktur repräsentieren, ist Aufgabe der *Dialogsteuerungsebene*[12].

[11] Modelliert durch W. Meyer und dem Autor im Zuge des *UCUI*-Projektes [2].
[12] Erläuterung folgt in Kapitel 4.

3.2. Weltmodell

Die Änderung des Parameters einer inneren physischen Komponente ist bei bestimmten Anwendungen von weiteren Parametern abhängig. Diese Abhängigkeit kann durch den Systementwickler entweder explizit modelliert oder eliminiert werden. Die entwickelte Verhaltenssteuerung ist in der Lage, unvollständige Spracheingaben[13] als bgMWR-Graph perzeptionsseitig zu akzeptieren und anschließend eine Frage als bgMWR-Zeichenkette an den Nutzer zu generieren und bisherige Teilbedeutungen miteinander zu kombinieren. Damit das System erkennt, ob Informationen zur Änderung eines Parameters fehlen, gleicht es zur Laufzeit den aktuell zu verarbeitenden bgMWR-Graph mit dem Weltmodell ab. Dieses repräsentiert verschiedene semantische Anker und ihre Abhängigkeiten voneinander. Das kann einerseits die Abhängigkeit von zwei semantischen Ankern sein, die in direkter Beziehung zueinander stehen oder über einen gemeinsamen Vorgängerknoten verfügen (indirekte Beziehung). In der entwickelten Verhaltenssteuerung benötigt zum Beispiel die Heizungssteuerungsebene zur Änderung der Thermostattemperatur einen Ausführungszeitpunkt und jede Fragestellung einen konkreten subjektiven Realitätsbezug. Die Einträge im Weltmodell sind dafür zum Beispiel:

Kategorie	vorhandener sem. Anker	geforderter sem. Anker
LVL:	*HEATABS*, *HEATREL*	*DATETIME*
CHILD:	*REQUEST*	_[14]

Des Weiteren existieren Einträge, die auf *Generalisierungen* verweisen. Diese verwendet das System zur Erzeugung einer allgemeineren Erwartungshaltung bezüglich des generierten bgMWR-Graphen. Der Eintrag dazu sieht zum Beispiel wie folgt aus

Kategorie	Generalisierung	zugeordnete sem. Anker
GEN:	*(HEAT)*	*HEATABS*, *HEATREL*

Durch die Einführung von *Generalisierungen* erkennt das System bei einem Vergleich von $G_2(bgmwr_2)$ und der zu $G_1(bgmwr_1)$ generierten Erwartungshaltung $G_E(bgmwr_E)$, ob es sich um das gleiche Nutzerziel handelt. In der vorliegenden Arbeit ist dies der Fall, wenn $G_2(bgmwr_2)$ zu $G_E(bgmwr_E)$ *kompatibel* ist. Das bedeutet: besitzt ein Knoten der Erwartungshaltung $v^c_{a,E} \in G_E(bgmwr_E)$ auf der Ebene $l \in \mathbb{N}$, eine endliche Menge von Nachfolgern $\Gamma_> \left(v^c_{a,E}\right)$, müssen alle semantischen Anker der Nachfolger $\Gamma_> \left(v^c_{a,2}\right)$ vom Knoten $v^c_{a,2} \in G_2(bgmwr_2)$ auf der gleichen Ebene, eine Teilmenge von semantischen Ankern der Knoten in $\Gamma_> \left(v^c_{a,E}\right)$ sein. Generalisierungen werden dabei durch die konkreten semantischen Anker ersetzt. Der Vorgang ist beispielhaft in Abbildung 16 dargestellt.

[13] Eingaben, die nicht den Systemanforderungen genügen, um einen Aktuatorparameter zu verändern.
[14] _ repräsentiert einen Platzhalter für einen unbekannten semantischen Anker.

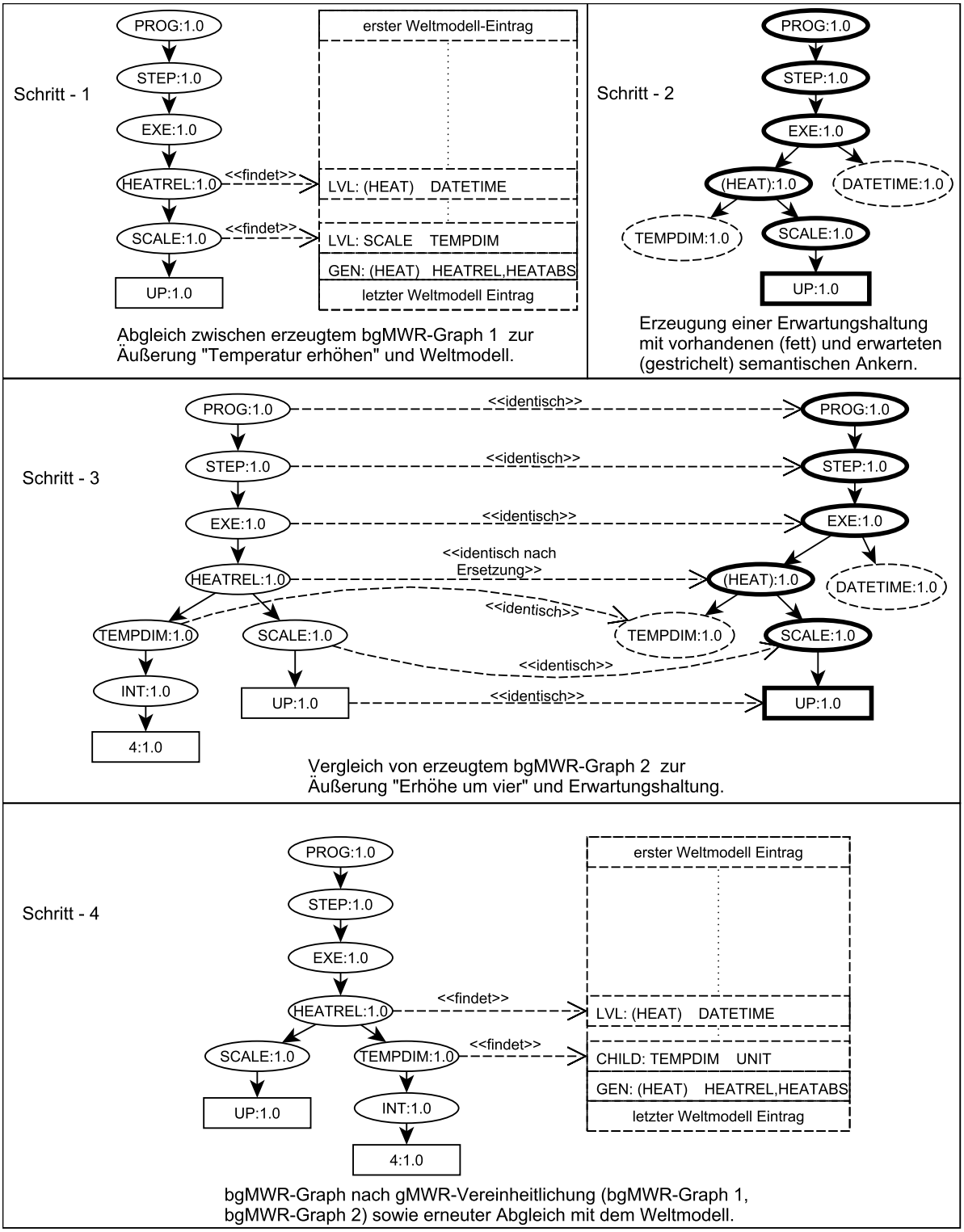


Abbildung 16: Beispiel zur Erzeugung der Erwartungshaltung zum bgMWR-Graphen 1 und der daraufhin folgenden bgMWR-Vereinheitlichung mit bgMWR-Graph 2 im nächsten Dialogschritt. Der Benutzer fordert im ersten Dialogschritt die Erhöhung der Temperatur, ohne den konkreten Wert zu nennen.

Das System vergleicht in einem ersten Schritt jeden semantischen Anker von $G_1(bgmwr_1)$ mit den Einträgen im Weltmodell und generiert anschließend mit den dort geforderten semantischen Ankern sowie der bekannten Struktur des bgMWR-Graphen $G_1(bgmwr_1)$ die Erwartungshaltung $G_E(bgmwr_E)$[15]. Kann das System keine Erwartungshaltung zu dem aktuell zu bearbeitenden bgMWR-Graphen erstellen, handelt es sich um eine ausführbare Systemaktion, welche an die Heizungssteuerungsebene weitergeleitet wird. Existiert jedoch eine Erwartungshaltung $G_E(bgmwr_E)$ zu dem aktuell zu verarbeitenden bgMWR-Graphen $G_1(bgmwr_1)$, übergibt das System diese als Paar $\big(G_1(bgmwr_1)$, $G_E(bgmwr_E)\big)$ an den Dialogstatus (siehe Abschnitt 3.3.1) und generiert eine Systemantwort $G_A(bgmwr_A)$. In dieser Arbeit stellen wir die Annahme auf, dass jede Systemantwort, die als bgMWR-Graph vorliegt, durch den Bedeutungs-Äußerungs-Transduktor in eine Zeichenfolge überführt werden kann. Der im folgenden Dialogschritt empfangene bgMWR-Graph $G_2(bgmwr_2)$ wird anschließend mit den im Dialogstatus abgelegten *Teilbedeutung-Erwartungshaltung* Paaren (T-E Paare) bezüglich der Kompatibilität verglichen. Sind $G_2(bgmwr_2)$ und die Erwartungshaltung $G_E(bgmwr_E)$ kompatibel zueinander, erfolgt im weiteren Verarbeitungsprozess die bgMWR-Vereinheitlichung zwischen $G_1(bgmwr_1)$ und $G_2(bgmwr_2)$.

3.3. Informationsstatus

Der Informationsstatus unterteilt sich in der entwickelten Verhaltenssteuerung in einen Dialog- und Heizungsstatus. Während der Dialogstatus alle Informationen eines laufenden Dialogs sowie nicht abgeschlossene Dialoge verwaltet, beinhaltet der Heizungsstatus alle noch ausstehenden Aktionen der Heizungssteuerungsebene.

3.3.1. Dialogstatus

Das Systemverhalten ist bei der Ausführung oder Bestätigung einer Aktion, die mit der Änderung eines inneren Aktuatorparameters in Beziehung steht, vom aktuellen Dialogstatus abhängig. Dieser setzt sich aus den drei Kellerspeichern *Dialog-Kellerspeicher*, *Tmp-Dialog-Kellerspeicher* und *Aktions-Kellerspeicher* zusammen (siehe Abbildung 17). Der Dialog-Kellerspeicher beinhaltet *Teilbedeutung-Erwartungshaltung* Paare, die einen nicht abgeschlossenen Dialog (aus Systemsicht) repräsentieren. Erfüllt der aktuell zu verarbeitende bgMWR-Graph die Anforderungen des Weltmodells nicht, fügt das System beide bgMWR-Graphen als Paar dem Dialog-Kellerspeicher hinzu und generiert einen

[15]Der Algorithmus zur Erzeugung ist in Abschnitt 4.1.3 erläutert.

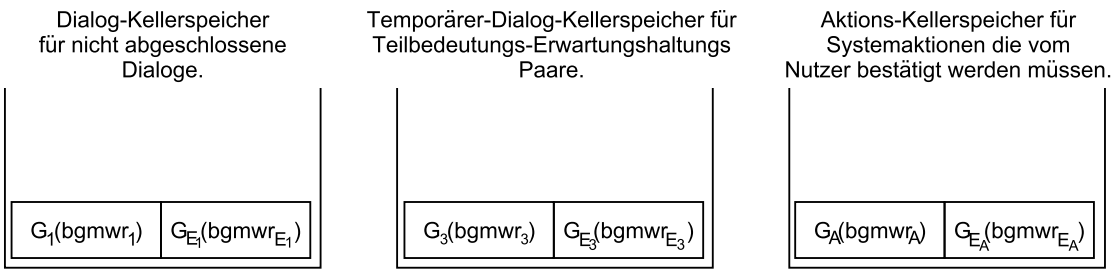


Abbildung 17: Komponenten des Dialogstatus mit beispielhaften Einträgen.

bgMWR-Graph mit einer Frage an den Nutzer. Besitzt der bgMWR-Graph im folgenden Dialogschritt einen semantischen Anker mit der Beschriftung *EXE*, vergleicht das System diesen mit der Erwartungshaltung des ersten Eintrags (Teilbedeutung-Erwartungshaltung Paare) vom Dialog-Kellerspeicher. Kann das System keine Kompatibilität zwischen dem neuen bgMWR-Graph und der Erwartungshaltung des Teilbedeutung-Erwartungshaltung Paares feststellen, durchsucht es den Dialog-Kellerspeicher solange, bis dieser leer ist. Der Ablauf ist durch die *while*-Schleife im Algorithmus 1 dargestellt.

Algorithmus 1 findComparableIntention $v_0(G_I)$

Data: dlgStack, tmpDlgStack

1: **while** dlgStack not empty **do**
2: $(v_0(G_{OI}), v_0(G_E))$:= dlgStack.pop()
3: isCompatible := **true**
4: isCompatible = compareAnchors$(v_0(G_I), v_0(G_E))$
5:
6: **if** isCompatible **then**
7: pushOldIntentionsBack$(dlgStack, dlgTmpStack)$
8: $v_0(G_N)$:= combineGraphs$(v_0(G_I), v_0(G_{OI}))$
9: checkSystemRequirements$(v_0(G_N), v_0(G_E))$
10: **return**
11: **else**
12: tmpDlgStack.push$((v_0(G_{OI}), v_0(G_E)))$
13: **end if**
14: **end while**
15: pushOldIntentionsBack$(dlgStack, tmpDlgStack)$
16: checkSystemRequirements$(v_0(G_I)$, null) ▷ null because no old user goal exist

Das System fügt jedes vom Dialog-Kellerspeicher genommene Teilbedeutung-Erwartungshaltung Paar, welches nicht mit dem aktuell zu verarbeitenden bgMWR-Graph kompatibel ist, dem Tmp-Dialog-Kellerspeicher hinzu und schreibt es bei einer erfolgreichen Übereinstimmung oder einem leeren Dialog-Kellerspeicher auf diesen zurück (vgl. Algorithmus 2). Dadurch können vorherige noch nicht abgeschlossene Dialoge zu einem späteren Zeitpunkt vom Benutzer fortgesetzt werden.

Algorithmus 2 pushOldIntentionsBack

Data: dlgStack, tmpDlgStack

```
1: while tmpDlgStack is not empty do
2:     dlgStack.push(tmpDlgStack.pop())
3: end while
```

Die hier entwickelte Verhaltenssteuerung ist somit in der Lage, den aktuellen Konversationsfokus anzupassen. Nach Abschluss eines Dialogs kann der Benutzer durch eine positive Bestätigung zum letzten nicht abgeschlossenen Dialog zurückkehren. Im entwickelten System kommt dieser Fall aufgrund des geringen Aktionsrepertoires nur selten vor. Mit der Erweiterung des Aktionsrepertoires sowie des Äußerungs-Bedeutungs-Transduktors entstehen jedoch weitere Abhängigkeiten, die während der Laufzeit vom Nutzer und dem System zu beachten sind.

Erkennt die Verhaltenssteuerung einen vollständigen bgMWR-Graphen[16], übergibt es diesen an die Heizungssteuerungsebene. Nach der dortigen Verarbeitung (siehe Abschnitt 4.2) speichert die Verhaltenssteuerung den verarbeiteten bgMWR-Graph in eine *Aufgabenliste*. Die dort hinterlegten Systemaktionen werden in Abhängigkeit vom gewählten Ausführungszeitpunkt zurück an die Dialogsteuerungsebene übermittelt und in den Aktions-Kellerspeicher gespeichert. Das System ist nun bestrebt, vom Nutzer eine Bestätigung bezüglich der auszuführenden Systemaktion zu erhalten und generiert dementsprechend eine Rückfrage an den Benutzer. Hier entsteht ein nebenläufiger Konflikt mit dem Dialog-Kellerspeicher. Während eines laufenden Dialoges darf das System deswegen nicht nach der Bestätigung zur Ausführung einer Systemaktion fragen. Dadurch kann es zur Verwirrung des Benutzers kommen, weil seine eigene Vorgeschichte[17] und die damit verbundene Erwartungshaltung gestört wird. Um diesen Zustand zu vermeiden, erfolgt nach dem Beenden eines Dialoges (Weitergabe des bgMWR-Graphen an die Heizungssteuerungsebene) die zeitlich verzögerte Abfrage des Aktions-Kellerspeichers. Besitzt dieser einen Eintrag, erzeugt die Verhaltenssteuerung eine Rückfrage und erwartet die Annahme oder Ablehnung der auszuführenden Systemaktion.
Das System verlangt ebenfalls eine positive oder negative Bestätigung hinsichtlich der Ausführung noch ausstehender Dialoge, die im Dialog-Kellerspeicher liegen. Vorausgesetzt, dass es sich im folgenden Dialogschritt um eine positive Bestätigung handelt, überprüft es zuerst die Erwartungshaltung des ersten Dialog-Kellerspeicher Eintrags und danach erst die des Aktions-Kellerspeicher Eintrags.

[16] Das System kann keine Erwartungshaltung erzeugen, weil alle Anforderungen des Weltmodells erfüllt sind.
[17] Ein Teil der Vorgeschichte des Benutzers bildet sich durch die Langzeitverwendung des Systems.

3.3.2. Heizungsstatus

Der Heizungsstatus besteht aus einer erweiterbaren Liste, deren Einträge aus einen Ausführungszeitpunkt und einem bgMWR-Graphen bestehen. Der bgMWR-Graph repräsentiert die auszuführende Aktion. Mit dem Erreichen des Ausführungszeitpunktes eines Eintrags sendet die Heizungssteuerungsebene den bgMWR-Graphen zur weiteren Verarbeitung an die Dialogsteuerungsebene und löscht den Eintrag aus der Aufgabenliste. Während der Programierung der Verhaltenssteuerungs-Komponente stellte sich die Frage, ob relative Sachverhalte in einem bgMWR-Graphen, die nicht den Ausführungszeitpunkt betreffen, bereits nach der Erkennung durch die Komponenten der Dialogsteuerungsebene bearbeitet werden sollen oder erst zum Ausführungszeitpunkt. Das damit verbundene Problem ist anhand des Beispiels zur Einstellung des Thermostatparameters mit späterem Ausführungszeitpunkt in Abbildung 18 dargestellt. In dem Beispiel wird das Nutzerziel durch die Beibehaltung des relativen Sachverhalts aus Systemsicht erfüllt. Aus Sicht des Nutzers trifft dies ebenfalls zu, wenn dieser über das Wissen bezüglich der Temperaturänderung zwischen k und $k+2$ verfügt. Ist dies nicht der Fall, bietet das System dem Benutzer zum Ausführungszeitpunkt $k+2$ die Ausführung einer Systemaktion an, welche nicht mit dem eigentlichen Nutzerziel übereinstimmt. Das System weiß jedoch nicht, ob der Nutzer bereits die Veränderung weiterer Parameter berücksichtigt.

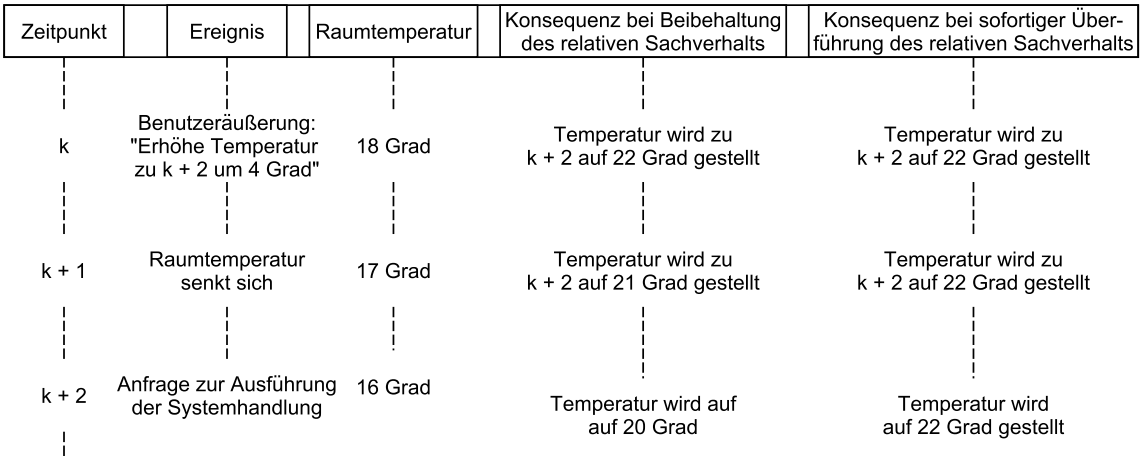


Abbildung 18: Beispiel zum Problem von relativen versus absoluten Sachverhalten, die zu einem späteren Ausführungszeitpunkt unterschiedliche Konsequenzen aufweisen. Es wird von einer Bestätigung der Systemhandlung zum Zeitpunkt $k+2$ ausgegangen.

Das entwickelte System überführt die relativen Angaben einer auszuführenden erst zum Ausführungszeitpunkt in absolute Angaben. In der Kombination mit einer Rückfrage zur Ausführungsbestätigung kann der Benutzer selber entscheiden, ob die relative Angabe noch immer gültig sein soll, ohne eine weitere Anfrage an das System zu stellen.

4. Verhaltenssteuerung - Kognitive Heizung

Das Verhalten des entwickelten Assistenzsystems mit kognitiver Nutzerschnittstelle ist nach den im letzten Kapitel getroffenen Aussagen vom aktuellen Informationsstatus (Dialog- und Heizungsstatus), dem eigenen Weltmodell und der internen Entscheidungslogik abhängig. Im vorliegenden Kapitel beschäftigen wir uns mit den Prozessen der Verhaltenssteuerung zur Laufzeit, den softwaretechnischen Strukturen innerhalb des Systems und den verwendeten Algorithmen zur Modifizierung eines bgMWR-Graphen (im Folgenden mit $G_{(Name)}$ gekennzeichnet). Aus softwaretechnischer Sicht kann die Verhaltenssteuerung in die bereits angesprochenen Ebenen – *Dialog-* und *Heizungssteuerungsebene* – unterteilt werden. Zum Austausch von Informationen dient eine Schnittstelle, welche beide Ebenen miteinander verbindet. Die Systemkomponenten der Verhaltenssteuerung sind in Abbildung 19 dargestellt.

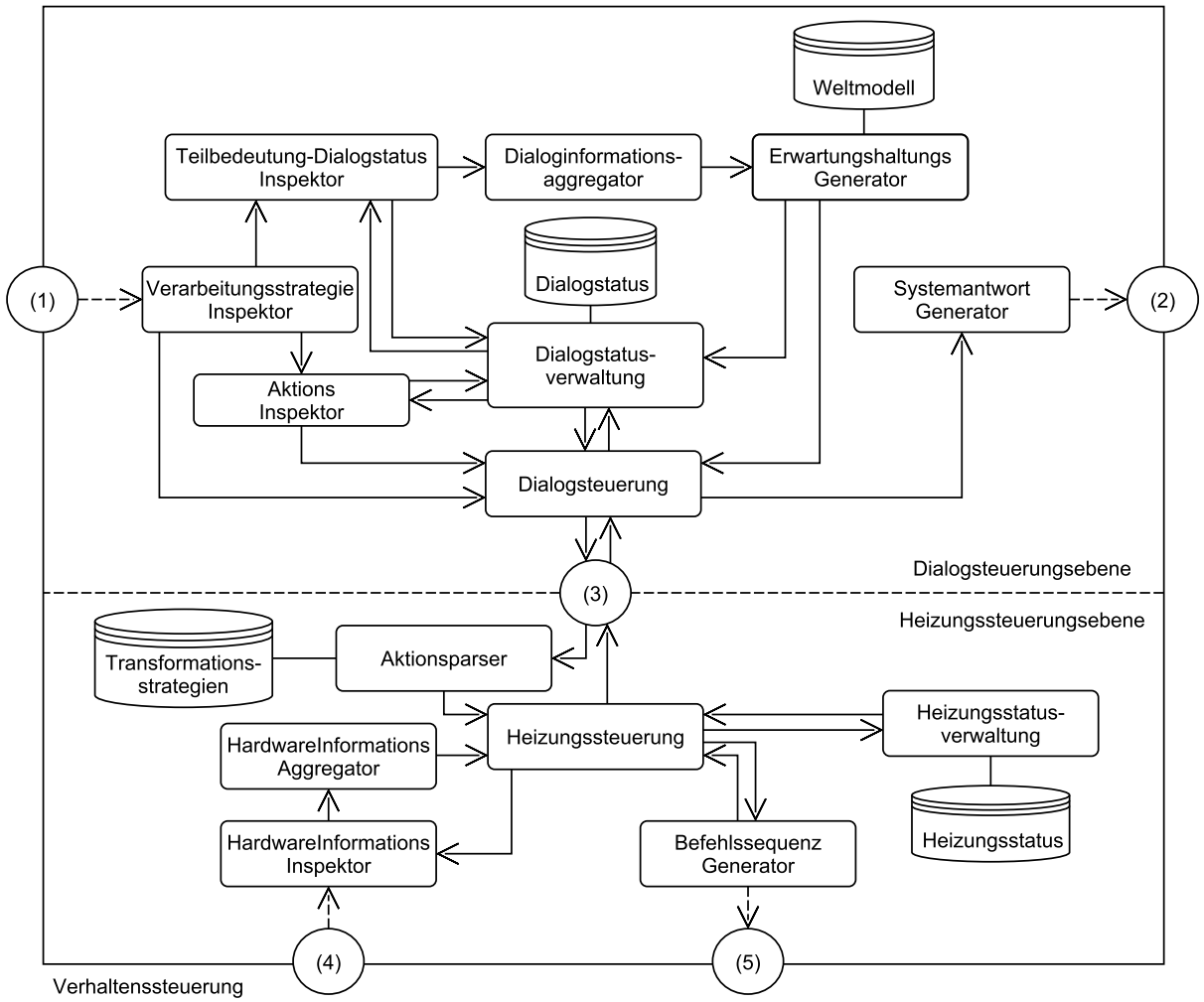


Abbildung 19: Systemkomponenten, Schnittstellen und Informationsfluss der Verhaltenssteuerung. Schnittstellen: (1) äußerer Perzeptionsbereich, (2) äußerer Aktionsbereich, (3) Dialog-Heizungsebene, (4) innerer Perzeptionsbereich, (5) innerer Aktionsbereich.

Die Implementierung der Verhaltenssteuerung erfolgte in das bereits vorhandene *CSL Softwareprojekt* (siehe [1]), welches zur Steuerung der physischen Komponenten im *Cognitive Systems Lab*[18] der Brandenburgischen Technischen Universität dient. Das *CSL Softwareprojekt* basiert auf der objektorientierten Programmiersprache *Java* und ist in *Packages* unterteilt. Die Struktur der für diese Arbeit relevanten *Packages* ist im Anhang A abgebildet.
In den folgenden Abschnitten werden keine Aussagen zu existierenden Peripherie-Systemen, wie zum Beispiel Laden des Weltmodells, getroffen. Diese sind ausschließlich im Anhang B als UML-Klassendiagramm dargestellt und durch eine *Javadoc* Dokumentation im Quellcode erläutert. Die entwickelte Verhaltenssteuerung arbeitet unabhängig von den anderen Komponenten im *CSL Softwareprojekt*. Dadurch kann eine Weiterentwicklung an der Verhaltenssteuerung erfolgen, ohne die Integrität des restlichen Projekts zu gefährden. Zur Verwendung der Verhaltenssteuerung ist die Klasse *BehaviorManager.java* als Instanz der Schnittstelle *ICognitionLauncher.java* zu laden. Diese gibt alle im Code Fragment 1 aufgelisteten Methodensignaturen zurück.

```
1 public IUserSpeechListener getUserSpeechListener();
2 public void setMachineListener(IMachineSpeechEvent listener);
```

Code Fragment 1: Signaturen der Schnittstelle ICognitionLauncher.java

Der Entwickler erhält über die Methode *getUserSpeechListener()* das Objekt der Verhaltenssteuerung, welches ein bgMWR-String in der eBNF-Form als Übergabeparameter erhält. Es handelt sich hier um die Schnittstelle des äußeren Perzeptionsbereiches. Für den Erhalt einer Systemantwort erstellt der Entwickler eine Klasse, die die Schnittstelle *IMachineSpeechEvent.java* implementiert und übergibt die Instanz zur Laufzeit an die Methode *public void setMachineListener(IMachineSpeechEvent listener)* der *BehaviorManager.java* Instanz. Es wird empfohlen, die Instanz des *BehaviorManagers* bereits während der Initialisierungsphase des *CSL Softwareprojektes* vorzunehmen.
Die Einbindung zusätzlicher innerer physischer Komponenten erfolgt durch die Programmierung weiterer Endgeräte-Klassen, welche *AInternalHardware.java* als Superklasse besitzen. Des Weiteren ist der Klassenname in die Textdateien *cognition.hardware.internal.txt* (Liste der zu ladenen Endgeräte-Klassen) und *cognition.hardware.SemCmd.txt* (Abbildung der Beschriftungen semantischer Anker auf die Namen der Endgeräte Klassen) hinzuzufügen. Es ist zu beachten, dass die entwickelte Verhaltenssteuerung ausschließlich für den spezifizierten Anwendungsfall – Heizungssteuerung – entworfen und getestet wurde. Die Änderung des Weltmodells und der damit verbundenen Ontologie kann somit zu nicht vorhersehbaren Fehlern führen.

[18] Labor zur experimentellen Untersuchung von akustischen und sprachverarbeitenden Systemen.

4.1. Dialogsteuerungsebene

In der vorliegenden Arbeit besteht die zentrale Aufgabe der Dialogsteuerungsebene in der Weiterverarbeitung einer empfangenen bgMWR-Zeichenkette, der Verwaltung laufender Dialoge, der Verarbeitung von auszuführendenr Systemaktionen und der Generierung von Informationen für den Nutzer. Des Weiteren regelt sie die Reihenfolge der auszuführenden Aktionen. Die von der Dialogsteuerungsebene zu verarbeitenden bgMWR-Zeichenketten sind durch den modellierten Äußerungs-Bedeutungs-Transduktor vorgegeben und besitzen mindestens einen semantischen Anker, mit dem der generierte bgMWR-Graph einer Verarbeitungsstrategie (*Ausführung*, *Anfrage* oder *Bestätigung*) zugeordnet werden kann. Im Folgenden betrachten wir ausschließlich bgMWR-Graphen, die zur Ausführung einer Aktion dienen und somit keine Kaskadierung nach [28] aufweisen.

4.1.1. Transformation bgMWR-Zeichenkette zu bgMWR-Graph

Das System überführt zunächst die vom äußeren Perzeptionsbereich empfangene bgMWR-Zeichenkette in einen bgMWR-Graphen G_I. Dieser liegt innerhalb des Softwaresystems als *Kompositum* vor (vgl. Abbildung 20) und wird durch die Klassen *Fvr.java* sowie *LeafNodeParser.java* erzeugt. Der Algorithmus zur Übersetzung durchsucht in einem ersten Schritt rekursiv die bgMWR-Zeichenkette nach eckigen Klammerpaaren, die zusammengehören. Ermittelt der Algorithmus ein Klammerpaar, ruft er die *parse* Methode auf und übergibt den aktuellen Knoten als Vorgängerknoten sowie die *Sub-Zeichenkette* zwischen den Klammern und erzeugt einen neuen Nachfolgerknoten. Ist kein weiteres Klammerpaar in der *Sub-Zeichenkette* vorhanden, wertet der Algorithmus die Subzeichenketten der einzelnen Knoten aus und ordnet ihnen eine Beschriftung (Name des semantischen Ankers) und einen Konfidenzwert zu. In einem zweiten Schritt werden jedem Knoten weitere Attribute hinzugefügt. Im weiteren Verlauf steht der Begriff semantischer Anker für die Beschriftung eines Knotens.

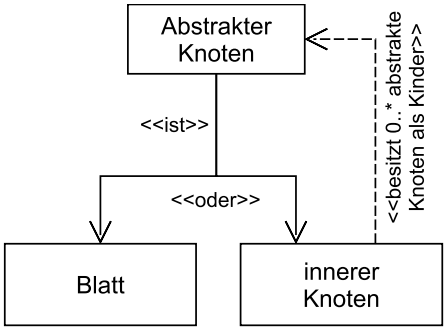


Abbildung 20: Blockdiagramm der Graphenstruktur innerhalb des Softwaresystems. Ein detailiertes Analyse-Klassendiagramm befindet sich im Anhang B.

4.1.2. Verarbeitungsstrategien

Mit der Auswahl einer Verarbeitungsstrategie trifft das System eine erste Entscheidung, die anschließend das gesamte Systemverhalten beeinflusst. Der *FindProcessingStrategy* Algorithmus überprüft zur Auswahl rekursiv die Beschriftungen der Knoten (ausgehend vom Startknoten) des erzeugten G_I. Stimmt die Beschriftung mit dem Namen eines Hauptschemata überein, wählt das System die zugeordnete Verarbeitungsstrategie aus.

Algorithmus 3 findProcessingStrategy $v(G_I)$

```
1: global variables
2:     s := null                                  ▷ initialized in meta algorithm
3: end global variables
4: if s not null then return s
5: end if
6: switch label(v(G_I)) do
7:     case REQUEST
8:         s = RequestStrategy
9:         break
10:    case CONFIRM
11:        s = ConfirmStrategy
12:        break
13:    case EXE
14:        s = ExecuteStrategy
15:        break
16: if |v(G_I).getChildren()| != 0 && s is null then
17:    {v'(G_I)} := v(G_I).getChildren()
18:    for each v'(G_I) ∈ {v'(G_I)} do
19:        findProcessingStrategy(v'(G_I))
20:    end for
21: end if
```

Frage-Strategie (RequestStrategy)

Das modellierte *Äußerungs-Bedeutungs*-Sprachmodell lässt nur vollständige Fragestellungen, ohne relative Sachverhalte, an die Verhaltenssteuerung zu. Dadurch müssen Prozesse wie Aktualisierung des Dialogstatus oder Transformation von einem relativen zu einem absoluten Sachverhalt anschließend nicht berücksichtigt werden. Das System führt somit nur die sequentielle Prozesskette: *bgMWR-Graph an Heizungssteuerungsebene senden* → *bgMWR-Graph an Kommando-Generator senden* → *Erwartungshaltung, Befehlssequenz generieren und versenden* → *Antwortzeichenfolge empfangen* → *Aggregation von Erwartungshaltung und erzeugtem bgMWR-Graph* → *Übermittlung des aggregierten bgMWR-Graphs an Dialogsteuerungsebene* → *Generierung der Systemantwort* aus.

Bestätigungs-Strategie (ConfirmStrategy)

Eine Bestätigung des Nutzers zu einem gegebenen semantischen Sachverhalt kann, wie in Abschnitt 3.3.1 angesprochen, entweder die Wiederaufnahme eines nicht abgeschlossenen Dialogs vom Dialog-Kellerspeicher oder die Ausführung einer Systemhandlung des Aktions-Kellerspeichers sein. Das Aktivitätsdiagramm in Abbildung 21 zeigt die verschiedenen Verhaltensweisen in Abhängigkeit vom Dialogstatus.

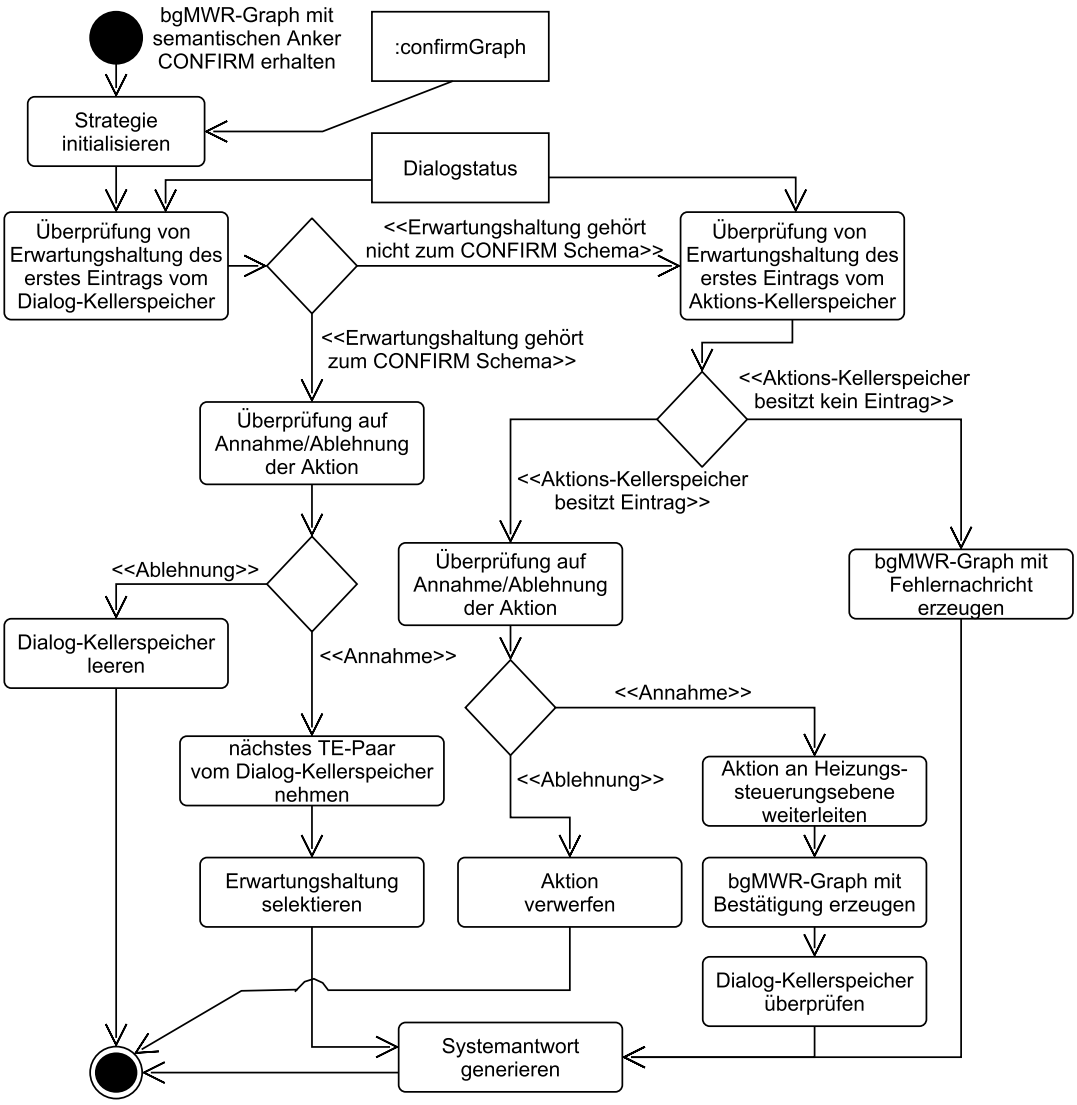


Abbildung 21: Aktivitätsdiagramm zum Ablauf der Bestätigungs-Strategie mit den auszuführenden Prozessen nach *UML2.0*-Standard (siehe [24]). <<[...]>> kennzeichnet die jeweilige Bedingung zur Ausführung des Zweiges.

Besitzt der erste Eintrag im Dialog-Kellerspeicher keine Erwartungshaltung, in der ein semantischer Anker mit der Beschriftung *CONFIRM* vorkommt, überprüft es den Aktions-Kellerspeicher. Existiert in diesem kein Eintrag, teilt das System dem Benutzer mit,

dass keine Aktionen zur Bestätigung existieren. Enthält der Dialog-Kellerspeicher jedoch einen Eintrag, leitet das System bei positiver Bestätigung die Aktion des Eintrags an die Heizungsebene weiter und generiert eine Systemantwort vom Schema *INFOEXE*. Anschließend prüft die Verhaltenssteuerung, ob sich nicht abgeschlossene Dialoge im Dialog-Kellerspeicher befinden. Ist dies der Fall, erzeugt das System ein Teilbedeutungs-Erwartungs Paar (die Erwartungshaltung beinhaltet den semantischen Anker *CONFIRM*) und fügt dieses dem Dialog-Kellerspeicher hinzu. Damit verbunden, fragt das System den Benutzer nach der Wiederaufnahme von nicht abgeschlossenen Dialogen. Unter der Annahme, dass es sich im folgenden Dialogschritt um eine negative Bestätigung zur Fortführung eines nicht abgeschlossenen Dialogs handelt, leert das System den gesamten Dialog-Kellerspeicher. Bei einer positiven Bestätigung erzeugt das System hingegen einen bgMWR-Graphen mit Fragen bezüglich der Erwartungshaltung des ersten nicht abgeschlossenen Dialoges.

Ausführungs-Strategie (ExecuteStrategy)

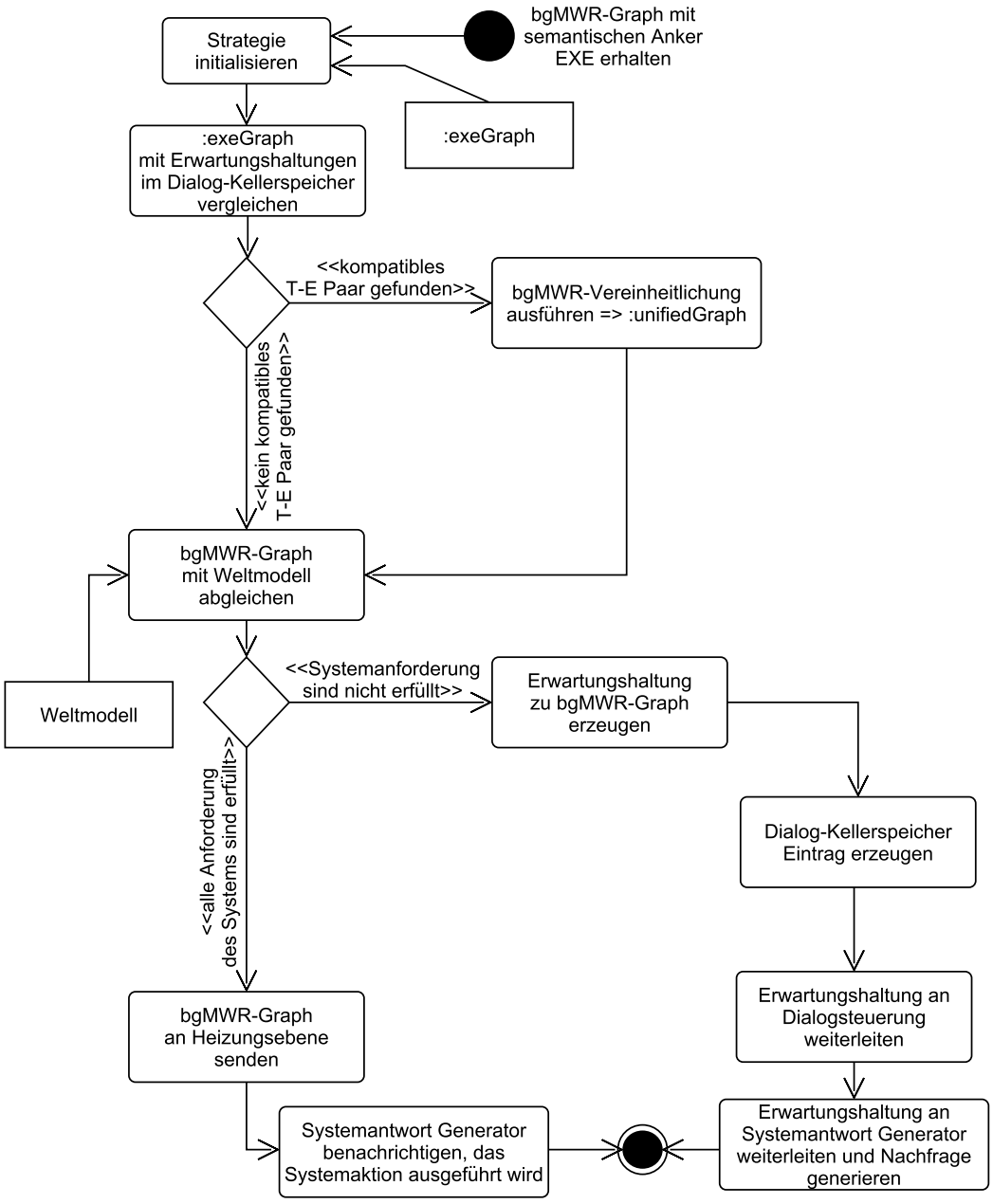
Enthält der erzeugte bgMWR-Graph einen semantischen Anker *EXE*, wählt das System die Ausführungsstrategie. Die Dialogsteuerungsebene vergleicht zunächst, wie in Abbildung 22 dargestellt, die Erwartungshaltungen der nicht abgeschlossenen Dialoge im Dialog-Kellerspeicher mit dem erzeugten bgMWR-Graphen. Dabei prüft es mit dem *compareAnchors* Algorithmus 4, ob die semantischen Anker auf einer Ebene des bgMWR-Graphen eine Teilmenge von sematischen Ankern der Erwartungshaltung sind[19].

Algorithmus 4 compareAnchors($v(G_I), v(G_E)$)

```
1: global variables
2:     isCompatible := true                          ▷ var in meta algorithm
3: end global variables
4: {v'(G_E)} := v(G_E).getChildren(), {v'(G_I)} := v(G_I).getChildren()
5: if {v'(G_E)} not empty and isCompatible then
6:     for each v'(G_I) ∈ {v'(G_I)} do
7:         if label(v'(G_I)) in {label(v'_||(G_E))} then
8:             compareAnchors(v'(G_I), v'_||(G_E))
9:         else
10:            isCompatible = false
11:            return
12:        end if
13:    end for
14: end if
```

[19]Der Algorithmus ist Bestandteil der Klasse *MeaningDialogStateInspector.java*.

Abbildung 22: Aktivitätsdiagramm der Ausführungs-Strategie.



Nimmt die globale Variable *isCompatible* im Algorithmus 4 den Wert *false* an, ist der bgMWR-Graph G_I nicht kompatibel zur Erwartungshaltung G_E. Das zum Vergleich verwendete Teilbedeutungs-Erwartungshaltungs Paar wird dem Tmp-Dialog-Kellerspeicher hinzugefügt (vgl. Algorithmus 1). Anschließend vergleicht das System G_I mit dem nächsten Eintrag des Dialog-Kellerspeichers. Ist der Variablenwert nach der Überprüfung *true*, erfolgt die Aktualisierung des Dialog-Kellerspeicher Eintrags.

4.1.3. Aktualisierung des Dialog-Kellerspeichers

Aggregation von Teilbedeutungen

Unter der Annahme, dass eine Kompatibilität zwischen dem aktuell zu verarbeitenden Graph G_I und der Erwartungshaltung G_E des Dialog-Kellerspeicher Eintrags (G_{OI}, G_E) vorliegt, vereinheitlicht das System die Graphen G_I und G_{OI}. Der Algorithmus 5 ist dabei die softwaretechnische Umsetzung der mathematischen Operation *bgMWR-Vereinheitlichen* aus Abschnitt 2.2.3. Der Algorithmus gilt jedoch nur unter der Bedingung, dass die Beschriftungen der Wurzeln beider bgMWR-Graphen gleich ist. Diese Voraussetzung ist jedoch in dieser Arbeit immer gegeben, weil alle durch den Äußerungs-Bedeutungs-Transduktor erzeugten Bedeutungen einen Wurzelknoten mit der Beschriftung *PROG* besitzen.

Algorithmus 5 combineGraphs$(v(G_I), v(G_{OI}))$

```
1: if label(v(G_I)) and label(v(G_OI)) not equal then        ▷ v(G_I) is another
2:     u(G_OI) := v(G_OI).getParent()                        part of generlization
3:     u(G_OI).removeChild(v(G_OI))
4:     u(G_OI).addChild(v(G_I))
5:     v(G_I).setParent(u(G_OI))
6:     return
7: else
8:     {v'(G_I)} := v(G_I).getChildren()
9:     {v'(G_OI)} := v(G_OI).getChildren()
10:    cnt := 0
11:    for each v'(G_I) ∈ {v'(G_I)} do                       ▷ find compatible node pairs
12:        if v'_||(G_OI) = findEqualAnchor(v'(G_I), {v'(G_OI))} not null then
13:            combineGraphs (v'(G_I), v'_||(G_OI))
14:            return
15:        else
16:            if v'(G_I).isLeaf() then
17:                for each v'(G_OI) ∈ {v'(G_OI)} do
18:                    if v'(G_OI).isLeaf() then
19:                        v(G_OI).removeChild(v'(G_OI))
20:                        break
21:                    end if
22:                end for
23:            end if
24:            v(G_OI).addChild(v'(G_I))                     ▷ add part of new
25:            v'(G_I).setParent(v(G_OI))                    graph to old graph
26:        end if
27:    end for
28: end if
```

Generierung der Erwartungshaltung

Zur Steuerung der inneren physischen Komponenten benötigt die Heizungssteuerungsebene verschiedene Informationen (vgl. Abschnitt 3.2). Erst wenn der bgMWR-Graph alle Anforderungen des Weltmodells erfüllt, handelt es sich um eine *ausführbare Systemaktion*. Damit die Verhaltenssteuerung erkennt, ob eine ausführbare Systemaktion vorliegt, vergleicht das System nach Algorithmus 6 jeden semantischen Anker des bgMWR-Graphen G_I mit den Einträgen (im Folgenden als *Regeln* bezeichnet) des Weltmodells[20]. Erfüllt G_I nicht alle für ihn zutreffenden Regeln, erzeugt der *Erwartungshaltungs Generator* eine neue Erwartungshaltung G'_E neben der alten kompatiblen Erwartungshaltung G_E.

Algorithmus 6 checkSystemRequirements($v(G_I), v(G_E)$)

```
1: global variables
2:     {rule} := worldModel.getRules()                          ▷ load rules
3:     replaceGeneralizations(), v0(G'_E) := null
4: end global variables
5: if !v(G_I).isLeaf() then                          ▷ no leafs because they were
6:     for each rule ∈ {rule} do                      inspected in last dialog step
7:         if label(v(G_I)) and rule.demandedAnchor() equal then
8:             ruleType := checkRuleType(rule)
9:             {v'(G_I)} := null
10:            if ruleType == LVL then
11:                u(G_I) := v(G_I).getParent()
12:                {v'(G_I} = u(G_I).getChildren()
13:            else
14:                {v'(G_I} = v(G_I).getChildren()
15:            end if
16:            if findCompatibleAnchor(rule, {v'(G_I)}) is null then
17:                if v0(G'_E) is null then                  ▷ no expection exist
18:                    v0(G'_E) = markedDeepClone(v0(G_I))
19:                end if                          ▷ combine old expection
                                                       with new expection
20:                v0(G'_E) = combineGraphs(cutted(v(G_I),   ▷ cutted: modify v(G_I) to
                              rule.requiredAnchor()),v0(G'_E))  chain of nodes with required
21:            end if                                          anchor as leaf
22:        end if
23:    end for
24:    {v'(G_I)} := v(G_I).getChildren()
25:    for each v'(G_I) ∈ {v'(G_I)} do          ▷ compare each child with world model
26:        checkSystemRequirements(v'(G_I), v(G_E))
27:    end for
28: end if
```

[20] Ausgeschlossen davon sind die Generalisierungen *GEN:*

Eine Regel kann in der vorliegenden Arbeit entweder vom Typ *LVL* oder *CHILD* sein. Die in Algorithmus 6 verwendete Methode *findCompatibleAnchor*$(\{v'(G_I)\}, rule)$ überprüft, ob ein semantischer Anker in der Menge $\{v'(G_I)\}$ den geforderten semantischen Anker der Regel *rule* besitzt und gibt diesen anschließend zurück. *markedDeepClone*$(v_0(G_I))$ kopiert zunächst den bgMWR-Graph G_I vollständig mit seiner *Kompositum Struktur* und setzt das Knotenattribut *Markierung* aller Knoten auf den Wert *true*. Dadurch kann der *Teilbedeutung-Dialogstatus Inspektor* im nächsten Dialogschritt zwischen den semantischen Ankern, die durch den Äußerungs-Bedeutungs-Transduktor hinzugefügt worden sind und denen der Erwartungshaltung, unterscheiden. Mit der Methode *cutted*$(v(G_I)$,*rule.requiredAnchor()*$)$ erzeugt das System zunächst eine Kopie des Graphen G_I . Das System fügt anschließend den geforderten semantischen Anker als Nachfolgerknoten oder Geschwisterknoten (Knoten auf derselben Ebene des fordernden Knotens) hinzu und löscht rekursiv alle anderen Knoten, die nicht auf einem direkten Weg zum Wurzelknoten $v_0(G_I)$ liegen. Das Löschen der peripheren Knoten entspricht der Operation *bgMWR-Löschen* in Abschnitt 2.2.3. Existiert nach der Terminierung des Algorithmus ein Graph G'_E, erzeugt die Verhaltenssteuerung einen Dialog-Kellerspeicher Eintrag (G_I, G'_E).

4.1.4. Aktualisierung des Aktions-Kellerspeichers

Der Aktions-Kellerspeicher enthält, wie in Abbildung 17 dargestellt, *Aktions-Erwartungshaltungshaltungs* Paare, wobei es sich um *auszuführende Systemaktionen* $G_A(bgmwr_A)$ handelt. Zur besseren Übersicht sind die Komponenten der Dialogsteuerungsebene noch einmal in Abbildung 23 dargestellt.

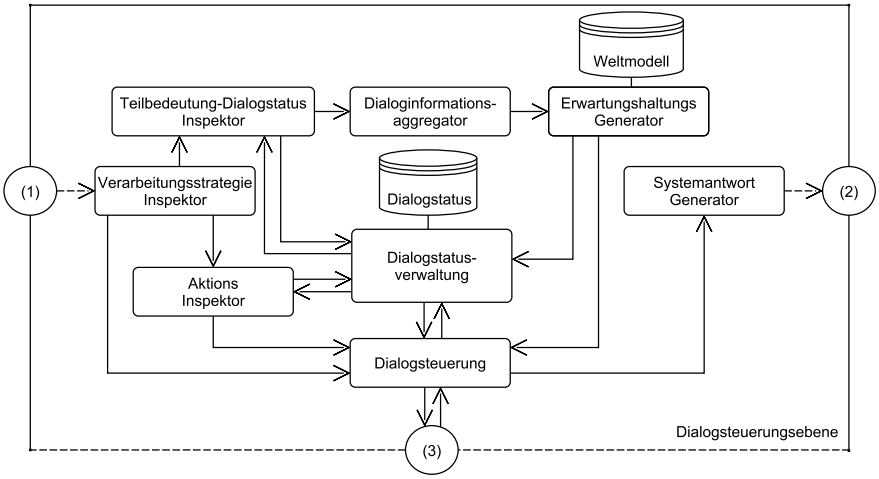


Abbildung 23: Systemkomponenten, Schnittstellen und Informationsfluss der Dialogsteuerungsebene. Schnittstellen: (1) äußerer Perzeptionsbereich, (2) äußerer Aktionsbereich, (3) Dialog-Heizungsebene.

Ein Eintrag in den Kellerspeicher kann ausschließlich über die Schnittstelle *IActionStackEntry.java* (vgl. Code Fragment 2) erfolgen, welche in den Klassen *DialogStateHandler.java* und *DialogController.java* implementiert ist und von der Klasse *HeatController.java* verwendet wird. Die Erzeugung der dazugehörigen Erwartungshaltung $G_{E_A}(bgmwr_{E,A})$ erfolgt statisch durch die Dialogstatusverwaltung und beinhaltet ausschließlich einen semantischen Sachverhalt zur Bestätigung.

```
1 public void newRunningAction(ANode entry);
```

Code Fragment 2: Signatur der Schnittstelle IActionStackEntry.java

Anschließend fügt das System den neuen Aktions-Erwartungshaltungshaltungs Eintrag dem Kellerspeicher hinzu und überprüft, ob aktuell ein Dialog zwischen Benutzer und System läuft. Dies geschieht über einen *lock-unlock* Mechanismus. Dabei fragt die *Dialogstatusverwaltung* ein Flag der *Dialogsteuerungs-Komponente* ab. Ist das Flag gesetzt, war die letzte Aktion der *Dialogsteuerungs-Komponente* ein Aufruf des *Systemantwort Generators* wegen eines nicht abgeschlossenen Dialoges. Das Flag wird negiert, wenn der *Erwartungshaltungs Generator* im nächsten Dialogschritt keine Erwartungshaltung an die *Dialogsteuerungs-Komponente* mitversendet oder ein vorgegebenes Zeitintervall für den Empfang weiterer Informationen vom *Erwartungshaltungs Generator* nicht eingehalten wurde. Mit der Negation erfolgt eine künstlich verzögerte Benachrichtigung der *Dialogstatusverwaltung*. Innerhalb der zeitlichen Verzögerung kann das System eine aktuell erkannte und zum nächstmöglichen Zeitpunkt auszuführende Systemaktion mit der dazugehörigen Erwartungshaltung ebenfalls dem Aktions-Kellerspeicher hinzufügen. Die *Dialogstatusverwaltung* prüft beim Erhalt der Benachrichtigung, ob diese durch das Beenden eines Dialoges oder mit Ablauf des Zeitintervalls in Verbindung steht. Unter der Annahme, dass der Dialog-Kellerspeicher keine weiteren Einträge enthält, nimmt das System den ersten Eintrag vom Aktions-Kellerspeicher (wenn vorhanden), sendet den bgMWR-Graph $G_A(bgmwr_A)$ über die *Dialogsteuerung* an den *Systemantwort Generator* und legt den Eintrag wieder zurück auf den Kellerspeicher. Empfängt die Verhaltenssteuerung anschließend einen bgMWR-Graph mit dem semantischen Anker *CONFIRM*, führt das System die *Bestätigungs-Strategie* nach Abschnitt 4.1.2 aus.
Mit der Rückfrage bezüglich des semantischen Sachverhalts – ist der aktuelle Zeitpunkt zur Ausführung der Systemaktion gewählt worden. Es stellt sich damit für den Entwickler die Frage, wie die Verhaltenssteuerung reagieren muss, wenn vom Benutzer keine Bestätigungsinformation über einen längeren Zeitraum vorliegt. Dies ist zum Beispiel der Fall, wenn der Benutzer zum Ausführungszeitpunkt nicht physisch anwesend ist. Die entwickelte Verhaltenssteuerung verwirft eine auszuführende Systemaktion, wenn der damit verbundene Dialog nicht in einem vorgegebenen Zeitraum bearbeitet wird.

4.2. Heizungssteuerungsebene

Die zentrale Aufgabe der Heizungssteuerungsebene liegt in der Bearbeitung sowie Verwaltung von Systemaktionen (bgMWR-Graphen) der Dialogsteuerungsebene. Des Weiteren transformiert sie die Systemaktionen zu Befehlssequenzen und bereitet die von den inneren physischen Komponenten empfangenen Informationen auf. Ein von der Dialogsteuerungsebene empfangener bgMWR-Graph ist aus Sicht der Heizungssteuerungsebene "vollständig" und kann durch die in Abbildung 24 dargestellten Systemkomponenten bearbeitet werden.

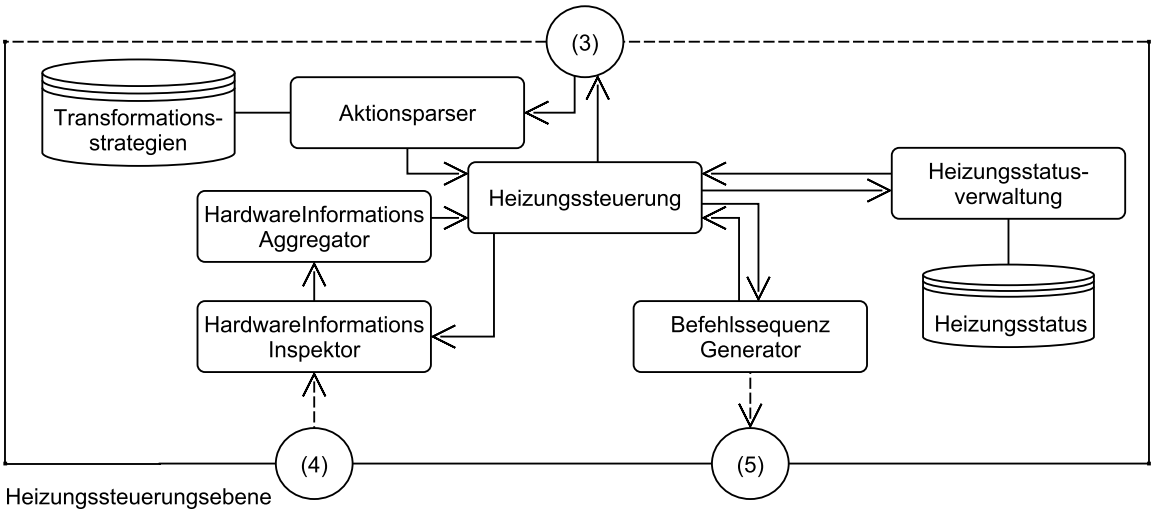


Abbildung 24: Systemkomponenten, Schnittstellen und Informationsfluss der Heizungssteuerungsebene. Schnittstellen: (3) Dialog-Heizungsebene, (4) innerer Perzeptionsbereich, (5) innerer Aktionsbereich.

Die *Dialogsteuerungs*-Komponente verwendet zur Übermittlung von Systemaktionen an die Heizungssteuerungsebene die Schnittstelle *INextHeatTask.java*, mit den im Code Fragment 3 dargestellten Signaturen. Die Heizungssteuerungskomponente *Aktionsparser* erkennt dadurch die zum empfangenen bgMWR-Graphen dazugehörige Verarbeitungsstrategie, ohne diesen noch einmal rekursiv überprüfen zu müssen.

```
1 public void newUserQuestion(ANode fvrGraph);
2 public void newSystemAction(ANode fvrGraph);
3 public void runSystemAction(ANode fvrGraph);
```

Code Fragment 3: Signaturen der Schnittstelle INextHeatTask.java

Handelt es sich bei dem bgMWR-Graphen um eine Frage bezüglich der inneren physischen Komponenten, wendet das System keine weitere Transformationsstrategie zur Überführung eines relativen Sachverhalts auf den bgMWR-Graph an. Das ist legitim, weil das in Abschnitt 3.1 beschriebene Sprachmodell ausschließlich Fragestellungen mit abso-

luten semantischen Sachverhalten zulässt. Handelt es sich jedoch um eine Systemaktion vom *Erwartungshaltungs Generator* (versendet über die *Dialogsteuerungs*-Komponente), überprüft die *Aktionsparser*-Komponente alle semantischen Anker auf relative semantische Sachverhalte, die mit dem Ausführungszeitpunkt in Verbindung stehen. Stammt der empfangene bgMWR-Graph hingegen ursprünglich vom *Aktions-Inspektor*, wendet das System eine Transformationsstrategie zur Überführung relativer Temperaturangaben an.

4.2.1. Transformation und Vervollständigung von relativen Sachverhalten

Die meisten inneren physischen Komponenten (Sensoren und Aktuatoren) können nur absolute Angaben ausgeben beziehungsweise verarbeiten. Diese Annahme stellen wir ebenfalls für die inneren physischen Komponenten der Heizungssteuerung auf. Alle relativen semantischen Sachverhalte sind deshalb vor der weiteren Verarbeitung durch die in Abbildung 24 dargestellte *Aktionsparser*-Komponente in einen absoluten semantischen Sachverhalt zu transformieren. Dafür „verrechnet“ die Verhaltenssteuerung den relativen Sachverhalt mit einem zum Zeitpunkt k geltenden absoluten Sachverhalt durch eine Transformationsstrategie. Wie im vorherigen Abschnitt beschrieben, ist der Transformationszeitpunkt eines relativen Sachverhalts vom Verarbeitungsstatus abhängig. Eine konkrete Transformationsstrategie besitzt die Schnittstelle *ITransformationStrategy.java*. Die *Aktionsparser*-Komponente durchsucht den bgMWR-Graphen nach bekannten semantischen Ankern und wählt anhand dieser eine Strategie aus. Anschließend trennt (*Trennstelle*) sie den bgMWR-Graphen am Knoten des erkannten semantischen Ankers in zwei Teilgraphen auf und sendet den abgeschnittenen Graph an die konkrete Transformationsstrategie.

```
1 public ANode runTransformation(ANode fvrBranch);
```

Code Fragment 4: Signatur der Schnittstelle ITransformationStrategy.java

In Abhängigkeit der ausgewählten Transformationsstrategie generiert das System eine Anfrage an die inneren physischen Komponenten. Im Gegensatz zur Frage-Strategie beinhaltet die anschließende innere Erwartungshaltung nicht den semantischen Anker *INFOREQUEST*, sondern den semantische Anker *INFOSYSTEM*. Dadurch erkennt das System anschließend, dass es sich um eine Information für die Transformationsstrategie handelt. Diese liegt als absoluter Sachverhalt vor und wird zur Verrechnung an die laufende Transformationsstrategie gesendet. Abschließend fügt die *Aktionsparser*-Komponente das Ergebnis (bgMWR-Graph mit absoluten Sachverhalt) an der *Trennstelle* wieder an. Zur Verdeutlichung dieses Vorgangs sind die einzelnen Aktivitäten in Abbildung 25 dargestellt.

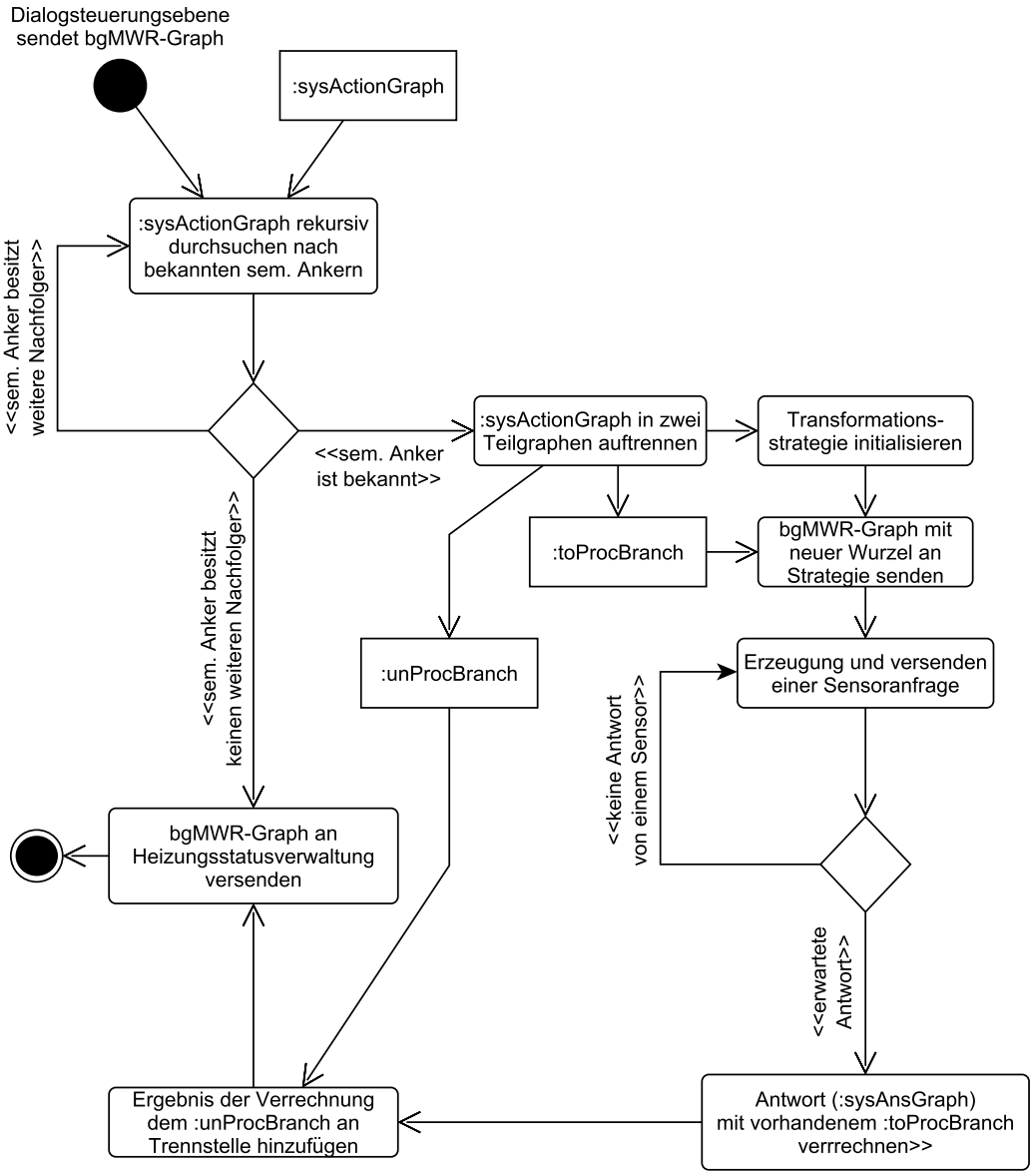


Abbildung 25: Aktivitätsdiagramm zum Ablauf eines Transformationsvorgangs.

Das Verfahren zur Verrechnung der relativen und absoluten Werte ist vom konkreten relativen Sachverhalt abhängig. Im Folgenden werden die Transformationsstrategien einer relativen Zeitangabe und die einer relativen Temperaturangabe erläutert.

Tranformation von Zeitangaben

Alle Angaben zum Ausführungszeitpunkt einer Systemaktion sind Bestandteil vom Teilschema *DATETIME*. Die Systemaktion kann relative oder absolute semantische Sachverhalte (zum Beispiel Uhrzeitangabe absolut oder relativ - siehe Abbildung 26) beinhalten. Des Weiteren kann die Systemaktion durch das verwendete Sprach- und Weltmodell nur bestimmte Teilbedeutungen des Ausführungszeitpunktes enthalten. Damit die Verhaltens-

steuerung jedoch eine Systemaktion ausführen kann, benötigt sie eine absolute Datums- und Uhrzeitangabe in der Form: Jahr-Monat-Tag-Stunde-Minute. Die Transformationsstrategie *DateTimeTransformation.java* hat somit die Aufgabe, einen bgMWR-Graphen zu erzeugen, der einen absoluten und vollständigen semantischen Sachverhalt hinsichtlich des Ausführungszeitpunktes repräsentiert.

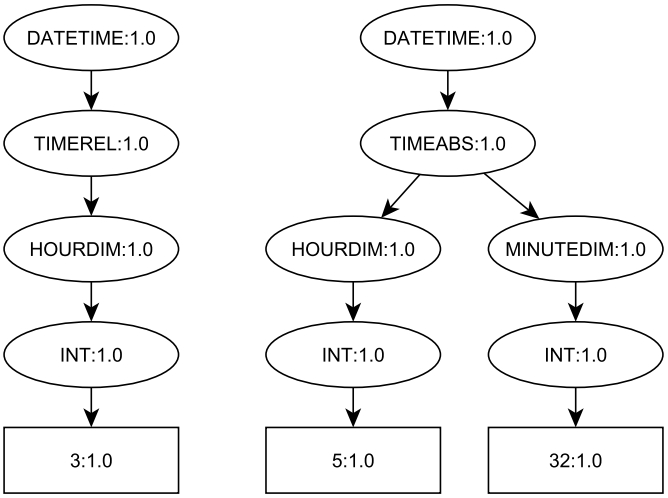


Abbildung 26: Semantische Anker zum Ausführungszeitpunt innerhalb einer Systemaktion. Sprachäußerung zur linken Teilbedeutung: „Stelle [...] in drei Stunden ein.“. Sprachäußerung zur rechten Teilbedeutung: „Stelle [...] um fünf Uhr zweiunddreißig ein“.

Die Transformationsstrategie erzeugt zunächst, wie in Abbildung 25 schematisch dargestellt, eine Anfrage an den internen *Uhrzeit* Sensor und erhält einen bgMWR-Graphen mit absoluten Zeitangaben zurück. Anschließend durchläuft das System die Teilbedeutung der Systemaktion rekursiv und speichert alle Zeitwerte mit einer Markierung in den vordefinierten zweidimensionalen Feldspeicher *timeValues* ab. Die hinzugefügte Markierung sagt aus, ob es sich um einen relativen oder absoluten Wert handelt. Besitzt ein Feld nach dem Vorgang keinen Wert, besaß der dazugehörige Zeitparameter keinen semantischen Anker im bgMWR-Graphen der zu verarbeitenden Systemaktion. Das System vergleicht für die Berechnung des neuen Ausführungszeitpunktes alle in *timeValues* abgespeicherten Zeitwerte mit denen der Sensorantwort. Die Generierung des neuen Ausführungszeitpunktes ist vom menschlichen Verständnis abhängig. Sagt ein Nutzer zum Beispiel „Stelle am Montag [...] ein.“ könnten folgende Ausführungszeitpunkte damit assoziiert werden:

- Systemaktion zur jetzigen Uhrzeit am nächsten Montag ausführen,
- Systemaktion zu unbestimmter Uhrzeit am nächsten Montag ausführen,
- Systemaktion um 0 Uhr am Montag ausführen oder
- Systemaktion zu bestimmten Zeitpunkt am Montag ausführen.

Besitzen die Speicherplätze *minute* und *hour* in *timeValues* keinen gültigen Wert, überschreibt das System die dazugehörigen semantischen Anker in der Systemantwort mit dem String *0*. Es wird somit Punkt drei für die Verarbeitung von Uhrzeitangaben angewendet. Datumsangaben verändert das System jedoch nicht bei ungültigen Werten im Feldspeicher.

Neben der variablen Interpretation von Zeitangaben muss das System ebenfalls Zeitübergänge beachten. Für die korrekte Verrechnung existiert innerhalb der Transformationsstrategie *DateTimeTransformation.java* die Methode *findSpecificAnchors*, welche die Sensorantwort (bgMWR-Graph) sequentiell nach den semantischen Ankern *MINUTEDIM*, *HOURDIM*, *OFWEEK*, *OFMONTH* sowie *MONTH* durchsucht. Erkennt das System einen spezifischen semantischen Anker, sind alle Nachfolger die Elemente von genau einem Pfad[21]. Erreicht der Algorithmus den letzten Knoten des Pfads, verrechnet er den Wert der Beschriftung mit dem des dazugehörigen Speicherplatzes in *timeValues*. Die zu verwendende Verrechnungsoperation ist dabei von der gesetzten oder nicht gesetzten Markierung abhängig. Nach der Verrechnung aller Zeitwerte gibt die Methode *runTransformation* der implementierten Schnittstelle *ITransformationStrategy.java* die angepasste Sensorantwort zurück an die *Aktionsparser*-Komponente.

Transformation von Temperaturangaben

Beinhaltet die auszuführende Systemaktion eine relative Temperaturangabe, ist diese durch die Verhaltenssteuerung in einen absoluten semantischen Sachverhalt zu überführen. Die Aussage „Erhöhe die Temperatur um vier Grad“ kann selbst in der Domäne Heizungssteuerung mehrere Bedeutungen besitzen. Zum Beispiel kann damit die Erhöhung der Raumtemperatur um vier Grad gemeint sein. Das System müsste dementsprechend den aktuellen Raumtemperaturwert abfragen, diesen um vier Grad erhöhen und anschließend den alten Thermostatwert mit dem Ergebnis überschreiben. Die Aussage könnte sich jedoch ebenfalls direkt auf den momentan geltenden Thermostatwert beziehen. In dem erstellten Sprachmodell bezieht sich die Äußerung auf die Erhöhung der aktuell geltenden Raumtemperatur, welche zum Ausführungszeitpunkt einen konkreten Wert besitzt.
Die vom *Aktionsparser* ausgewählte und erzeugte Instanz der Klasse *HeatTempTransformation.java* generiert zunächst einen bgMWR-Graph mit dem semantischen Sachverhalt zur Abfrage des Raumtemperatursensors. Anschließend übergibt das System die Instanz als Beobachter an die *Heizungssteuerungs*-Komponente. Der erzeugte bgMWR-Graph wird über die *Heizungssteuerung* an den *Befehlssequenz Generator* gesendet und die Erwartungshaltung konstruiert. Nach der Vereinheitlichung der eingehenden Ant-

[21] Diese Aussage gilt nur für das verwendete Sprachmodell.

wort – bgMWR-Graph zum Sensorwert – und der inneren Erwartungshaltung durch den *HardwareInformations Aggregator* sendet die *Heizungssteuerung* den bgMWR-Graph zur Instanz der Klasse *HeatTempTransformation.java* zurück. Die Verrechnung des absoluten und relativen Sachverhalts ist von verschiedenen Faktoren abhängig und zur besseren Übersicht in Abbildung 27 dargestellt.

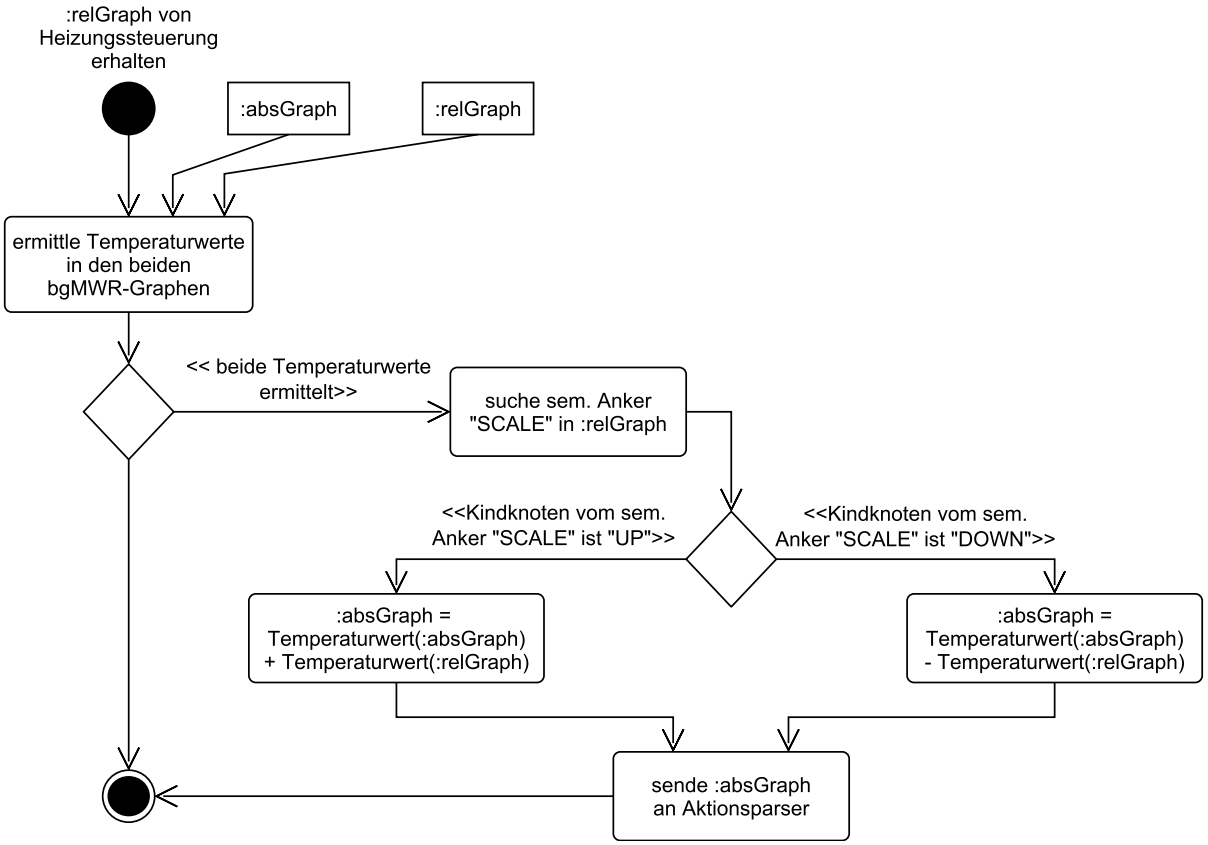


Abbildung 27: Ablauf der Temperaturverrechnung.

4.2.2. Aufgabenverwaltung

Nach der Transformation aller relativen Zeitangaben und deren Vervollständigung leitet die *Aktionsparser*-Komponente den modifizierten bgMWR-Graph über die *Heizungssteuerung* an die *Heizungsstatusverwaltungs*-Komponente weiter. Diese beinhaltet eine Aufgabenliste, welche den *Heizungsstatus* repräsentiert. Ein Listeneintrag besteht aus dem Tupel *(Ausführungszeitpunkt, auszuführende Systemaktion)*. Damit keine Komplikationen zwischen den Listeneinträgen entstehen, vergleicht die *Heizungsstatusverwaltungs*-Komponente zunächst den Ausführungszeitpunkt des empfangenen bgMWR-Graphs mit allen Ausführungszeitpunkten der vorhandenen Listeneinträge. Sind die Ausführungszeitpunkte zwischen bgMWR-Graph und Listeneintrag identisch, vergleicht

das System anschließend die auszuführenden Systemaktionen miteinander. Besitzen diese identische Merkmale, überschreibt das System die im Listeneintrag vorhandene auszuführende Systemaktion mit der neuen auszuführenden Systemaktion. Es existiert somit zu jedem spezifischen semantischen Sachverhalt nur eine auszuführende Systemaktion zu einem bestimmten Ausführungszeitpunkt.

Die *Heizungsstatusverwaltungs*-Komponente ist innerhalb des Softwaresystems ein Objekt der Klasse *HeatStateHandler.java* und besitzt die innere Klasse *HeatTaskControl*. Damit die Verhaltenssteuerung eine auszuführende Systemaktion für die Bestätigung zu ihrem Ausführungszeitpunkt an die Dialogsteuerungsebene sendet, ruft ein Timerobjekt periodisch die run-Methode der Klasse *HeatTaskControl* auf. Diese überprüft alle Ausführungszeitpunkte der Listeneinträge mit der aktuellen Uhrzeit und dem aktuellen Datum. Liegt der Ausführungszeitpunkt des Listeneintrags in der Vergangenheit oder ist er identisch zur aktuellen Uhrzeit-/Datumsangabe, sendet das System die auszuführende Systemaktion zur Bestätigung an die Dialogsteuerungsebene und löscht den Listeneintrag.

5. Bewertung und Ausblick

In der vorliegenden Masterarbeit sind nach Wissen des Autors erstmals beschriftete gewichtete Merkmal-Werte-Relation als Graphen für den Informationsfluss in einer Verhaltenssteuerung zum Einsatz gekommen und eine Algebra zum „Rechnen“ auf den semantischen Datenstrukturen definiert worden. Mit den erstellten Operationen *bgMWR-Hinzufügen*, *bgMWR-Löschen* sowie *bgMWR-Vereinheitlichung* war es möglich, die Algorithmen zur Modifizierung eines bgMWR-Graphen innerhalb der entwickelten Verhaltenssteuerung mathematisch zu beschreiben. Im Laufe dieser Arbeit stellte sich heraus, dass die entwickelte Verhaltenssteuerungs-Komponente in Kapitel 4 nicht nur zur vorgegebenen Aufgabe – Steuerung einer Heizung mittels Spracheingaben – verwendet werden kann, sondern zur universellen Steuerung von Maschinen, deren Sprachmodell auf den drei Metaschemata *REQUEST*, *EXE* und *CONFIRM* basiert. Zur Erweiterung des Funktionsumfangs sind die verwendeten Modelle anzupassen, eine neue Geräteklasse für die Ansteuerung der konkreten Hardware zu implementieren (vgl. Anhang B) und eventuell neue Strategien für die Übersetzung von relativen zu absoluten semantischen Sachverhalten hinzuzufügen. Neben den positiven Aspekten dieser Arbeit entstanden jedoch ebenfalls weiterführende Fragen und Probleme, die nicht abschließend behandelt werden konnten.

Annahme der idealen Spracherkennung

In der Realität ist die Spracherkennung immer mit Störungen verbunden. Die Annahme, dass alle Knoten eines übersetzten bgMWR-Graphen den Konfidenzwert 1 besitzen (ideale Spracherkennung), ist somit nicht zulässig. Des Weiteren kann das System zur Zeit nicht zwischen einer akustischen Störung oder einer unbekannten Spracheingabe nach der Aktivierung des Systems unterscheiden. Dementsprechend ist es nicht in der Lage, unterschiedliche Fehlerausgaben (Mitteilung an den Nutzer über unsichere Erkennung wegen akustischer Störung oder einer unbekannter Spracheingabe) zu erzeugen. Akustische Störgeräusche können mit Verfahren zur Störgeräuschunterdrückung korrigiert werden. Diese Verfahren sind Bestandteil der Audio- beziehungsweise Signal- sowie Sprachverarbeitung und in [3] und [29] ausführlich beschrieben. Zudem existieren Verfahren zur automatischen Fehlererkennung und -behebung für die Spracherkennung (siehe [16]). Ein weiterer Lösungsansatz[22] ist das Setzen eines Schwellwertes über die Konfidenzwerte aller Knoten in Abhängigkeit von der Knotenanzahl. Ist der durchschnittliche Konfidenzwert unter dem Schwellwert, entscheidet sich die Verhaltenssteuerung für eine Sprachausgabe mit dem semantischen Sachverhalt: unbekannte Eingabe.

[22] Alle genannten Lösungsansätze sind erste Ideen und nicht mathematisch betrachtet oder konkret ausformuliert. Das Ergebnis der Lösungsansätze ist somit nicht bekannt.

Erkennung von unbekannten Spracheingaben

Der für die Übersetzung einer Zeichenfolge in eine bgMWR-Zeichenkette zuständige Äußerungs-Bedeutungs-Transduktor übersetzt ausschließlich vorher festgelegte Zeichenketten in die dazugehörigen bgMWR-Zeichenketten. Das entspricht natürlich nicht der geforderten intuitiven Bedienung des Systems mittels Sprachäußerungen. Es ist an dieser Stelle jedoch zu erwähnen, dass bekannte kommerzielle Systeme ebenfalls nur vorher festgelegte oder erlernte Sprachäußerungen verarbeiten können. Unter der Annahme einer idealen automatischen Spracherkennung kann für die Erkennung der Zeichenfolge zur Sprachäußerung ein *Phonem N-Gram Modell* (Spracherkenner ohne Grammatik) zum Einsatz kommen. Für die Erkennung von unbekannten Spracheingaben wird der *äußerer Perzeptionsbereich* und die Dialogsteuerungsebene der *Verhaltenssteuerung* mit den in Abbildung 28 markierten Systemkomponenten erweitert.

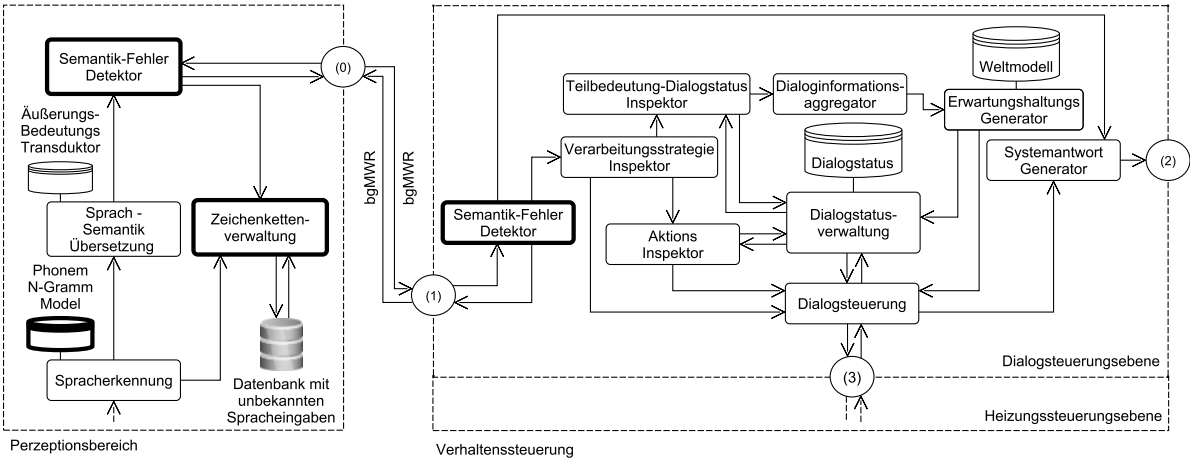


Abbildung 28: Ausschnitt des Systemmodells mit Erweiterung des äußeren Perzeptionsbereiches und der Verhaltenssteuerung zur Erkennung von unbekannten Spracheingaben.

Die Datenbank im äußeren Perzeptionsbereich dient zur Speicherung von *(Zeichenfolgen, bgMWR-Zeichenketten)* Einträgen. Diese legt das System bei einer unbekannten Spracheingabe an und teilt dem Benutzer anschließend mit, dass keine Aktion zugeordnet werden kann. Erfolgt in den folgenden Dialogschritten eine Spracheingabe, die durch den Äußerungs-Bedeutungs-Transduktor in eine bgMWR-Zeichenkette übersetzt wird, fragt das System rückwirkend nach einer Zuordnung der erkannten bgMWR-Zeichenkette zu der unbestimmten Spracheingabe. Bestätigt der Benutzer die Zuordnung, ist das System anschließend in der Lage, die unbekannte Spracheingabe in der Verhaltenssteuerung weiterzuverarbeiten. Für die Umsetzung des Lösungsansatzes ist jedoch die interne Entscheidungslogik des Systems anzupassen und der Umgang mit Konfidenzwerten zu klären.

Variabilität des Äußerungs-Bedeutungs-Transduktors

Der modellierte Äußerungs-Bedeutungs-Transduktor besteht aus einem endlichen Zustandsautomaten mit einem Start- und Endzustand. Beim Übersetzungsprozess können ausschließlich durchgehende Pfade des Automaten vom Start- zum Endzustand durchlaufen werden. Zur Zeit kann der Äußerungs-Bedeutungs-Transduktor dementsprechend keinen korrekten semantischen Sachverhalt zu einer Spracheingabe wie zum Beispiel „21 Grad“ generieren. Das Problem kann jedoch durch das Einfügen von *Epsilonkanten* (vgl. [12]) in den modellierten Äußerungs-Bedeutungs-Transduktor behoben werden. Eine Epsilonkante besitzt in unserem Fall das leere Zeichen als Eingabe- und Ausgabesymbol. Jeder innere Zustand des Äußerungs-Bedeutungs-Transduktors erhält eine Epsilonkante zur Verbindung mit dem Startzustand und eine weitere zur Verbindung mit dem Endzustand des Automaten. Der Transduktor kann dadurch zur Laufzeit eine endliche Anzahl von Zuständen und Kanten, mit den zugeordneten Ein- und Ausgabesymbolen, vom Startknoten aus überspringen. Die ermittelte bgMWR-Zeichenkette besitzt somit ausschließlich die semantischen Anker, welche in einer direkten Beziehung zu der erkannten Zeichenfolge – erzeugt durch die automatische Spracherkennung – stehen.

Selektion der Verarbeitungsstrategie

Mit der Einführung von Epsilonkanten, um jeden inneren Zustand des Äußerungs-Bedeutungs-Transduktors vom Startzustand zu erreichen, ist jedoch die Dialogsteuerungsebene der Verhaltenssteuerung neu zu modellieren. Diese Feststellung folgt bereits aus der Tatsache, dass die semantischen Anker *REQUEST*, *EXE* und *CONFIRM* mit dem oben beschriebenen Verfahren nicht mehr in einer bgMWR-Zeichenkette vorkommen müssen. Diese sind jedoch notwendig, um das interne Verarbeitungsverhalten auszuwählen. Es ist deshalb ein Algorithmus zu entwickeln, der alle möglichen – dem System bekannten – bgMWR-Graphen als Erwartungshaltung konstruiert, die mit dem unvollständigen bgMWR-Graphen kompatibel sind. Anschließend kann der Benutzer zu den Erwartungshaltungen befragt werden. Die Auswahl der Erwartungshaltung für die Befragung ist entweder zufällig oder durch einen Adaptionsalgorithmus zu erlernen.
Weiterhin soll in Zukunft das Systemverhalten nicht von den Beschriftungen einzelner Knoten abhängig sein, sondern von den Konfidenzwerten des Weltmodells. Dieses ist dem System zur Laufzeit als zusammenhängender gerichteter Graph bekannt. Erfolgt vom Benutzer eine Spracheingabe, ändern sich die Konfidenzwerte verschiedener semantischer Anker im Weltmodell. Das System schließt daraus auf eine Aktion und somit implizit auf die Bedeutung der Spracheingabe.

Weiterverarbeitung verketteter Systemaktionen

Neben der Annahme einer idealen Spracherkennung gingen wir in dieser Arbeit ausschließlich von atomaren Systemaktionen als Spracheingabe aus. Ein bgMWR-Graph konnte also nur eine einzige auszuführende Systemaktion als semantischen Sachverhalt repräsentieren. Diese Einschränkung ist nicht komfortabel für den Benutzer und somit aufzulösen. Zunächst sind in dem Äußerungs-Bedeutungs-Transduktor *Schleifen* einzufügen. Dadurch kann er mehrere Systemaktionen aus einer getätigten Spracheingabe erfassen. Des Weiteren ist der nach [28] im *CSL-Projekt* implementierte *Kaskadierungs*-Algorithmus zur Übersetzung der bgMWR-Zeichenkette in einen bgMWR-Graphen, zu verwenden. Die anschließende Weiterverarbeitung hängt von dem entwickelten Sprachmodell ab. Das in dieser Arbeit verwendete Sprachmodell fügt alle semantischen Anker, die zu einer Systemaktion gehören, unter den semantischen Anker *STEP* hinzu. Enthält ein bgMWR-Graph mehrere Knoten mit der Beschriftung *STEP*, können die einzelnen Systemaktionen in der Verhaltenssteuerung durch einen Vorverarbeitungsschritt, wie zum Beispiel in Abbildung 29 dargestellt, voneinander getrennt werden.

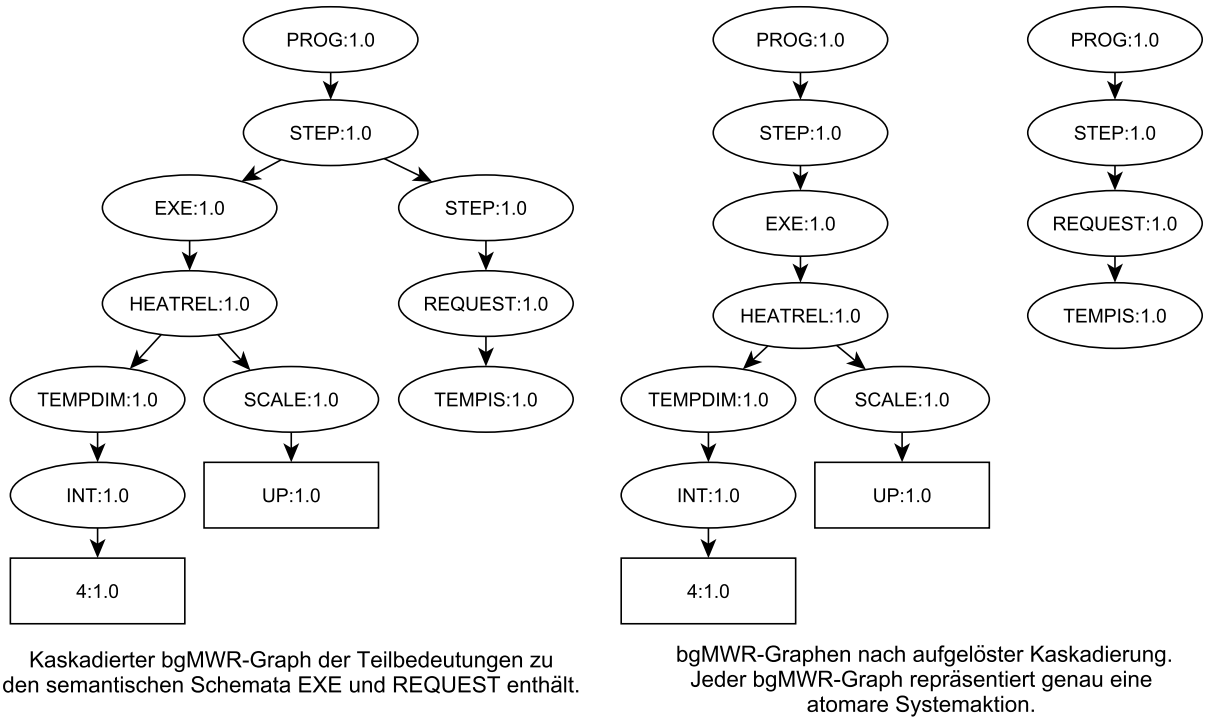


Abbildung 29: Trennung der Aktionen eines kaskadierten bgMWR-Graphens. Sprachäußerung: „Erhöhe die Raumtemperatur um 4 Grad und sage mir wie hoch die aktuelle Temperatur ist“.

Die Verarbeitung der einzelnen Aktionen kann anschließend sequentiell in der Kaskadierungsreihenfolge stattfinden. Der Lösungsansatz gilt jedoch nur, wenn die Aktionen innerhalb des bgMWR-Graphs konsequent voneinander getrennt sind.

Adaption des Nutzerverhaltens

Ein technisches dynamisches kognitives System ist in der Lage, aus den getätigten Aktionen und den sich daraus ergebenden Zuständen (Beobachtung der Umwelt) eine Verhaltensstrategie zu erlernen. In der betrachteten Anwendung können drei unterschiedliche Lernprozesse zum Einsatz kommen:

- Adaption an unbekannte Spracheingaben,
- Adaption an das sprachliche Nutzerverhalten und
- Adaption an gewählte Aktuatoreinstellungen.

Die Adaption unbekannter Spracheingaben des Nutzers ist bereits unter dem Punkt *Erkennung unbekannter Spracheingaben* diskutiert worden. Durch die Adaption an das sprachliche Nutzerverhalten kann das System unvollständige Nutzereingaben[23] einer Systemaktion automatisch zuordnen. Dafür greift die Verhaltenssteuerung auf einen episodischen Speicher zu, in dem die einzelnen bgMWR-Graphen der Dialogschritte einem konkreten *Dialog-Dialogschritte Tupel* zuordnet sind. Kennzeichnet das System einen *Dialog-Dialogschritte Tupel* im episodischen Speicher als abgeschlossen, kann dieser anschließend für einen Vergleich mit dem aktuell laufenden Dialog verwendet werden. Eine erste Anpassung der hier entwickelten Dialogsteuerungsebene sowie den sich daraus ergebenen Aktivitäten für die Ausführungs-Strategie sind in den Abbildungen 30 und 31 dargestellt.

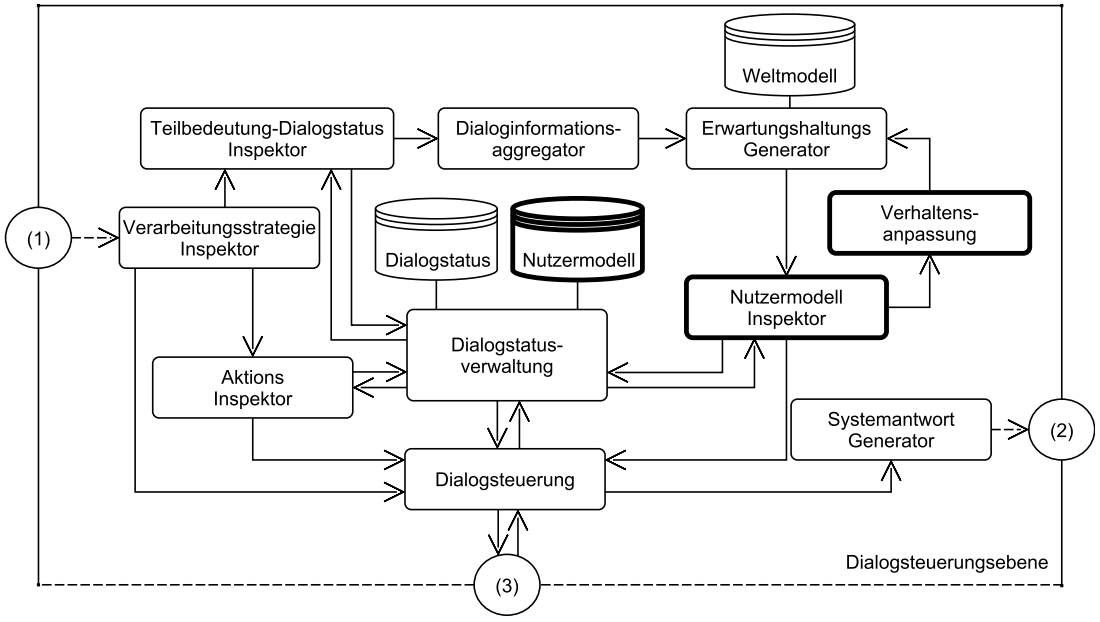


Abbildung 30: Erweiterung der Dialogsteuerungsebene für die Adaption des sprachlichen Nutzerverhaltens. Fett markierte Systemkomponenten sind neu hinzugekommen.

[23] Spracheingaben, die der Äußerungs-Bedeutungs-Transduktor übersetzen kann, jedoch nicht den Anforderungen des Weltmodells genügen.

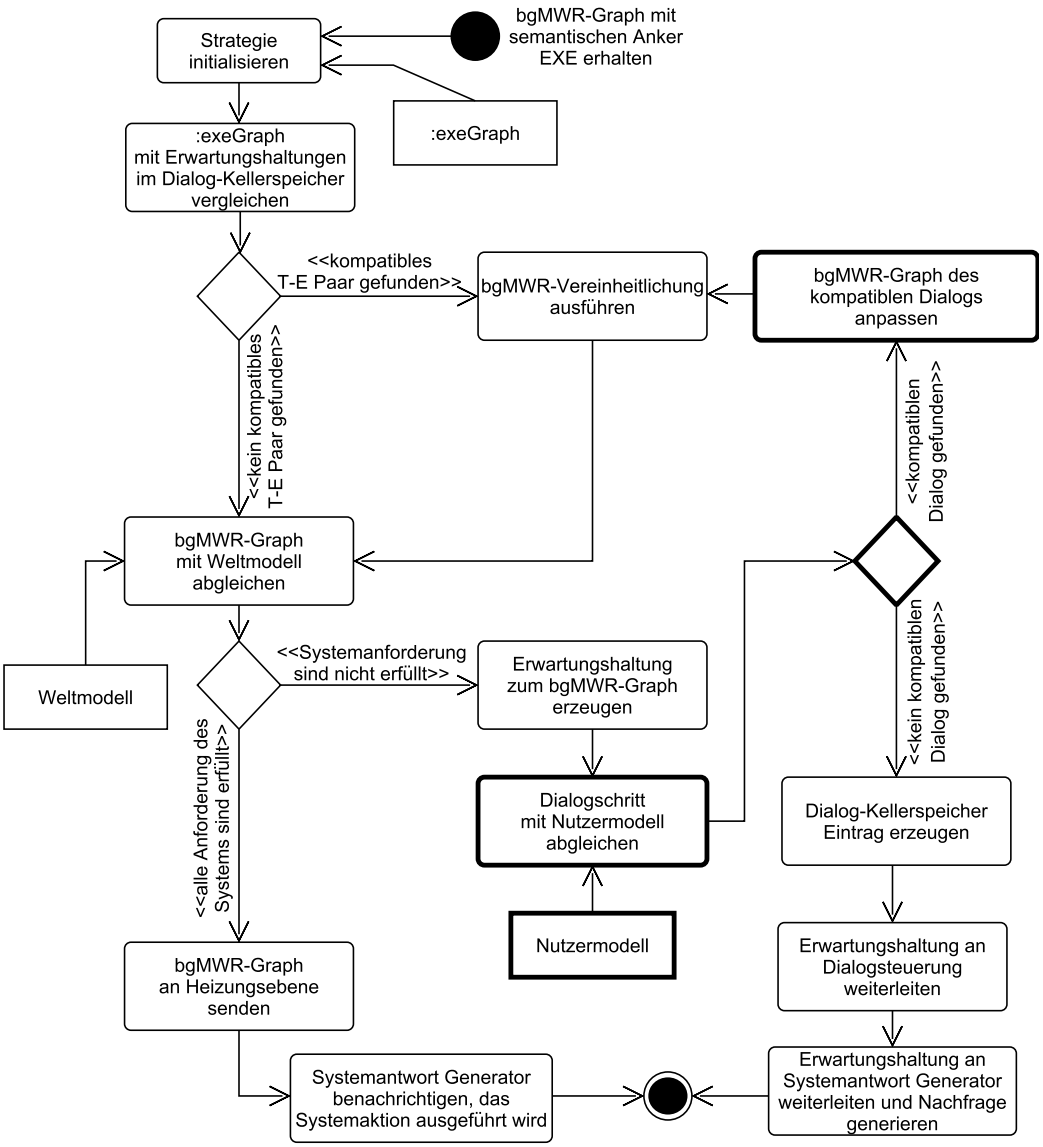


Abbildung 31: Aktivitätsdiagramm zum Ablauf der Ausführungs-Strategie innerhalb der Dialogebene unter der Einbeziehung der Adaption des sprachlichen Nutzerverhaltens

Mit der Einbindung des Lernprozesses ergeben sich jedoch weitere Fragen hinsichtlich der Anpassung des Nutzermodells, der Auswahl und Weiterverarbeitung einer Systemaktion oder der Veränderung des Systemverhaltens. Es ist zum Beispiel zu klären, ob die Auswahl automatisch vom System zu treffen ist oder zunächst eine Rückfrage an den Nutzer: „Meinten Sie . . . “ erfolgt.

Technische dynamische kognitive Systeme zeichnen sich in verschiedenen Literaturstellen (vgl. [10], [23]) dadurch aus, dass sie selbstständig Entscheidungen treffen. In der entwickelten Anwendung *Kognitive Heizung* könnte das System zum Beispiel automatisch die Thermostateinstellungen zu einer bestimmten Jahreszeit verändern, wenn der Benutzer in dieser immer einen bestimmten Temperaturwert bevorzugt hat und keine an-

deren Systemaktionen vorliegen. Dafür ist die Heizungssteuerungsebene jedoch *proaktiv* zu modellieren. Die Verhaltenssteuerung erhält dadurch in einem festgelegten Zeitintervall alle aktuellen Sensorwerte als bgMWR-Graph und vergleicht sie mit den erfassten Sensorwerten (Datum, Temperatur) von bereits ausgeführten Systemaktionen. Diese sind ebenfalls in einem episodischen Speicher hinterlegt. Bei der Entscheidungsfindung stellt sich jedoch ebenfalls die Frage, wie ein sinnvoller Vergleich von Sensorwerten der bereits ausgeführten Systemaktion und aktueller Sensorwerte erfolgen kann.

Fazit

Das Forschungsgebiet der technischen dynamischen kognitiven Systeme gewann in den letzten Jahren zunehmend an Relevanz. Die sich aus dieser Arbeit ergebenen Fragestellungen bezüglich einer kognitiven Gerätesteuerung verdeutlichen, dass die Erforschung des Gebietes gerade erst am Anfang steht und nicht ausschließlich technische, sondern auch ethische und psychologische Aspekte beachtet werden müssen. Diese sind wiederum vom konkreten Einsatzbereich des Systems abhängig. Vor allem die im letzten Abschnitt beschriebene autonome Ausführung von Systemaktionen ohne die direkte Einbeziehung des Benutzers ist kritisch zu sehen.

Die vorliegende Arbeit lieferte einen ersten Beitrag zur Modellierung und Implementierung einer Kognitiven Gerätesteuerung. In der weiteren Entwicklung sind die angesprochenen Erweiterungsvorschläge zu konkretisieren und umzusetzen. Parallel dazu ist ein Simulator zu entwickeln, in dem die Verhaltenssteuerungs-Komponente ausführlich getestet werden kann.

A. Paketstruktur des Softwareprojekts

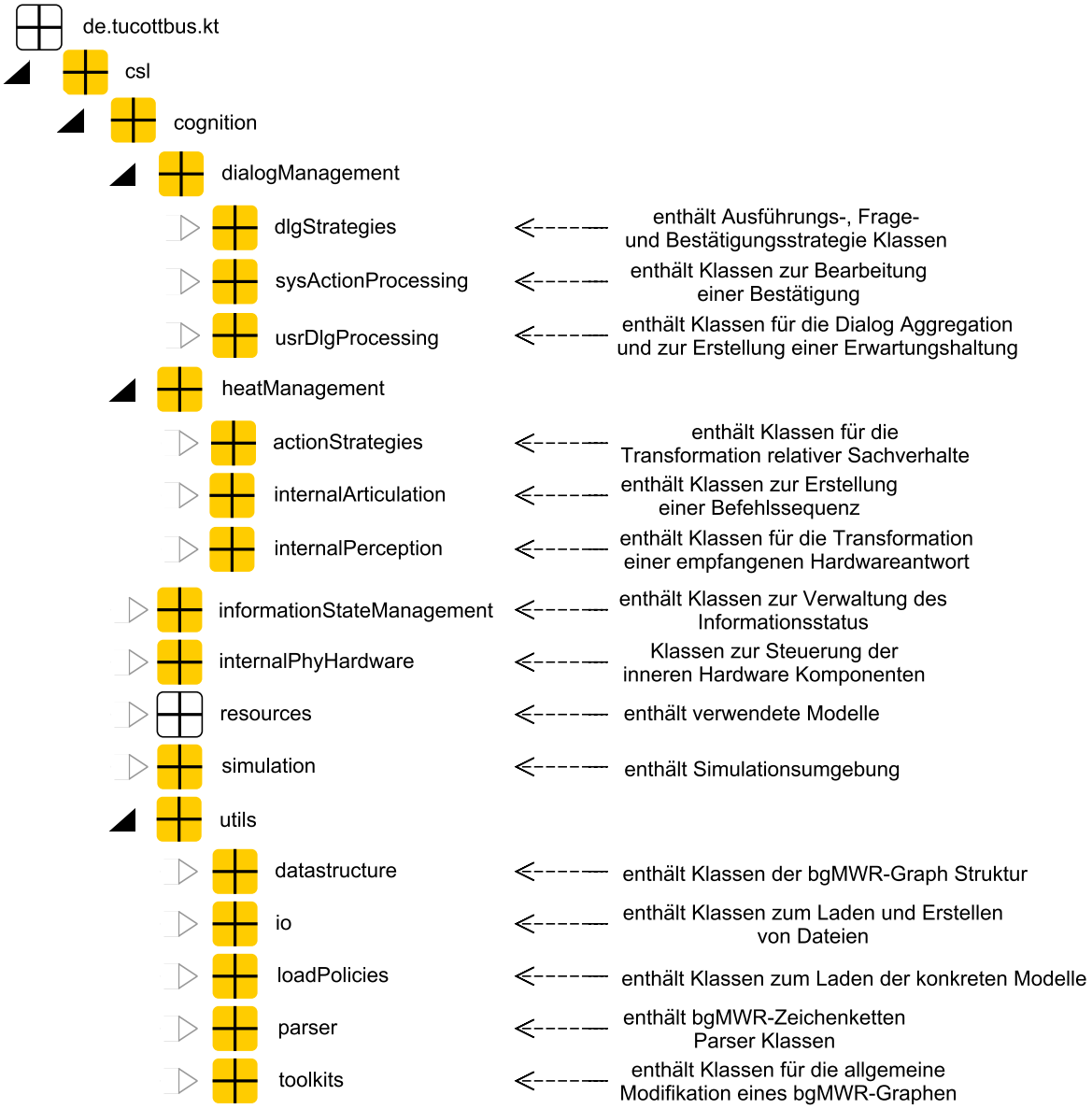


Abbildung 32: Unterteilung des Softwareprojekts in Pakete. Jedes Paket besitzt Klassen für ein bestimmtes Aufgabengebiet.

B. UML-Diagramme

B.1. Analyse-Klassendiagramm: bgMWR-Graph

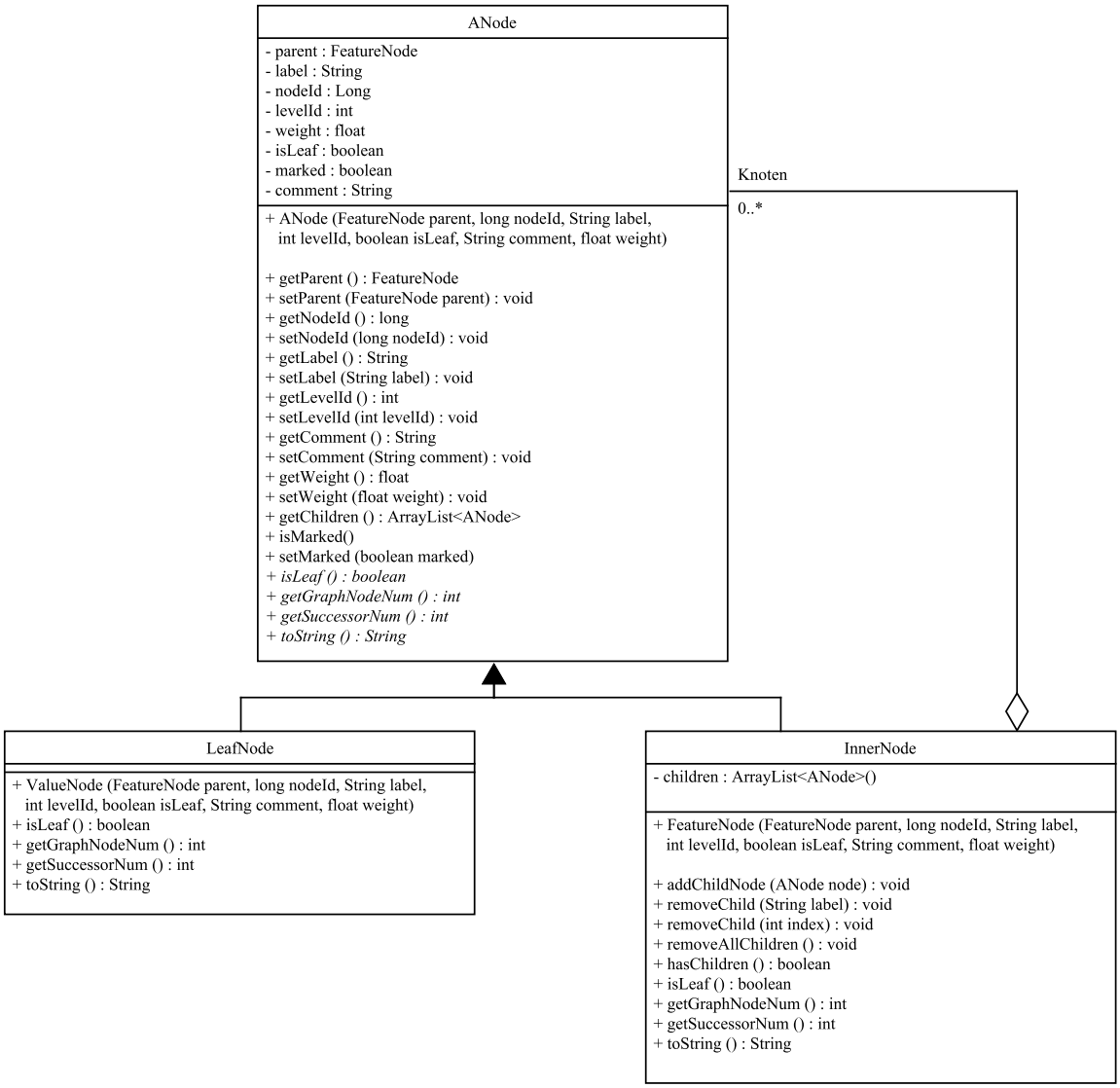


Abbildung 33: Detaillierte Struktur eines bgMWR-Graphen als *Kompositum*.

B.2. Analyse-Klassendiagramm: Interne Hardwareverwaltung

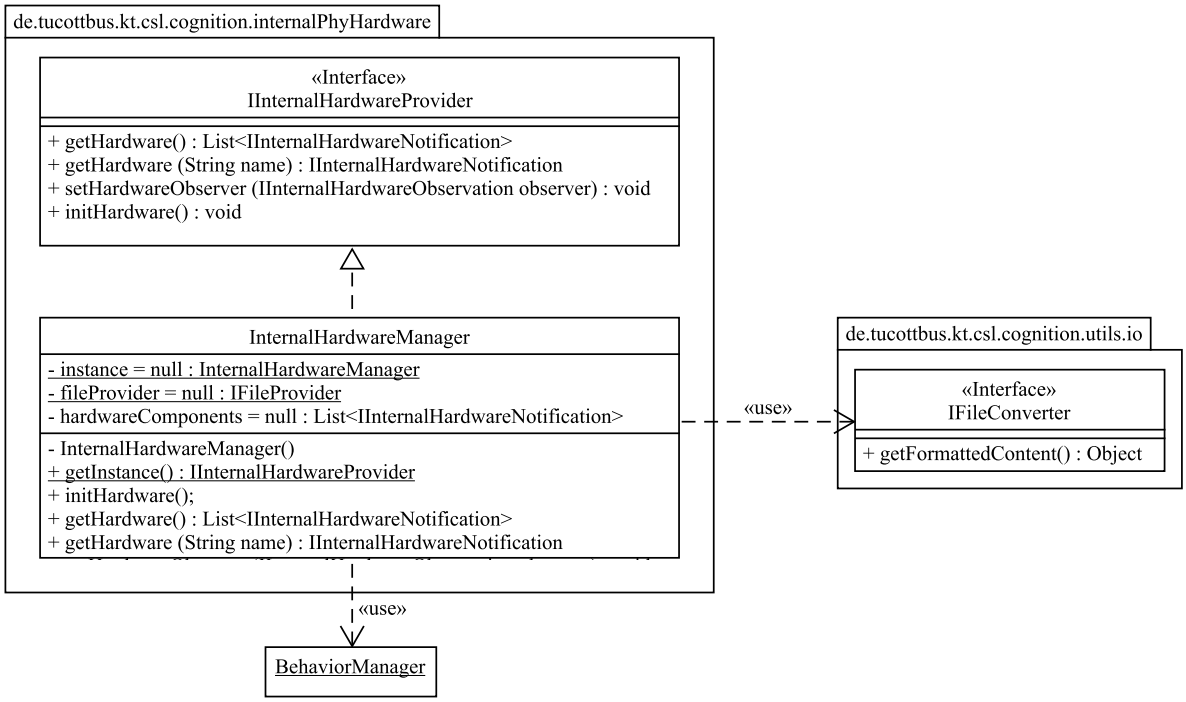


Abbildung 34: UML-Klassendiagramm zum Laden und abrufen der internen Hardware-Komponenten.

B.3. Analyse-Klassendiagramm: Aufbau der internen Hardware

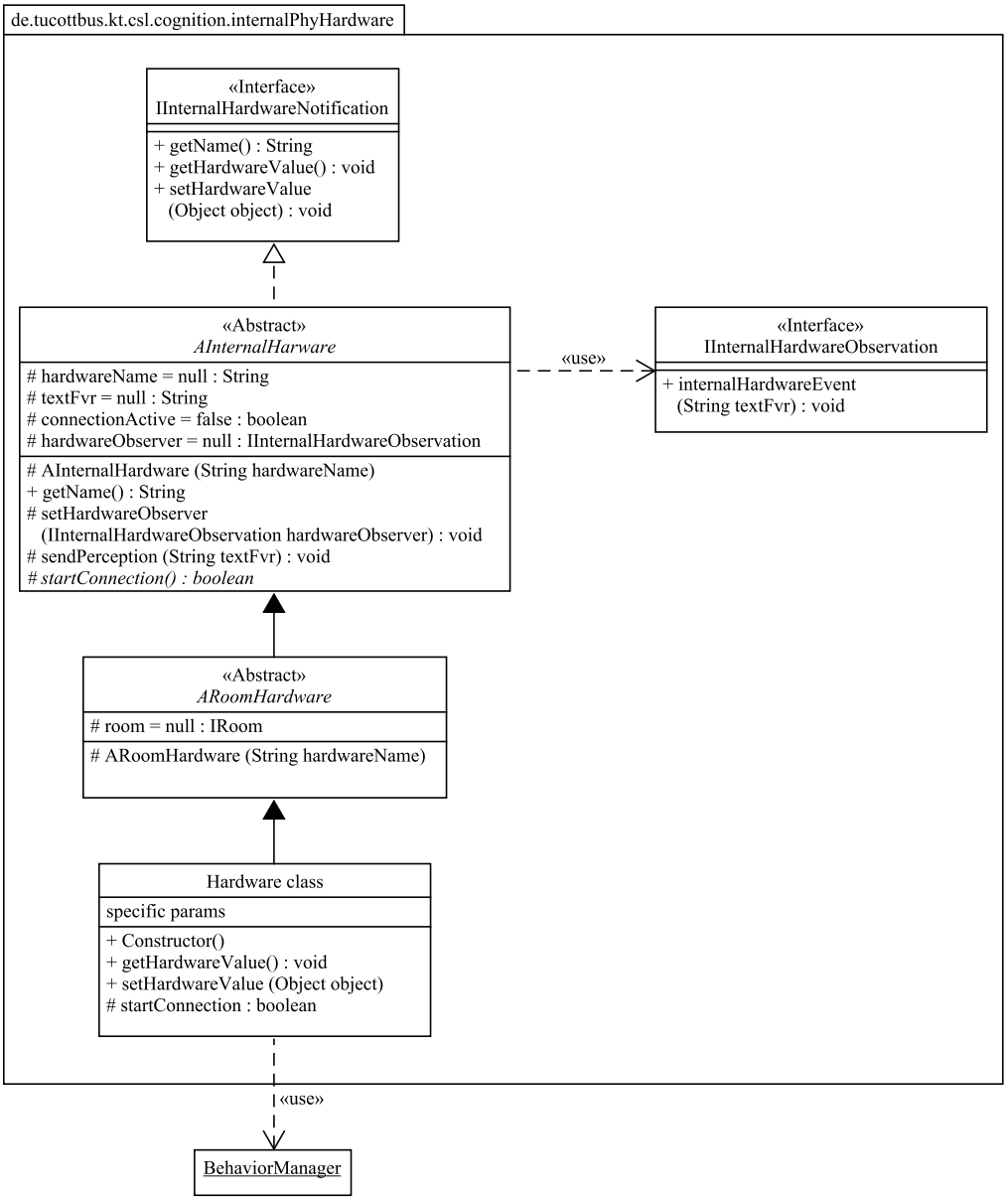


Abbildung 35: UML-Klassendiagramm zum Aufbau einer internen Hardware-Komponenten Klasse.

B.4. Analyse-Klassendiagramm: Laden der Modell Dateien

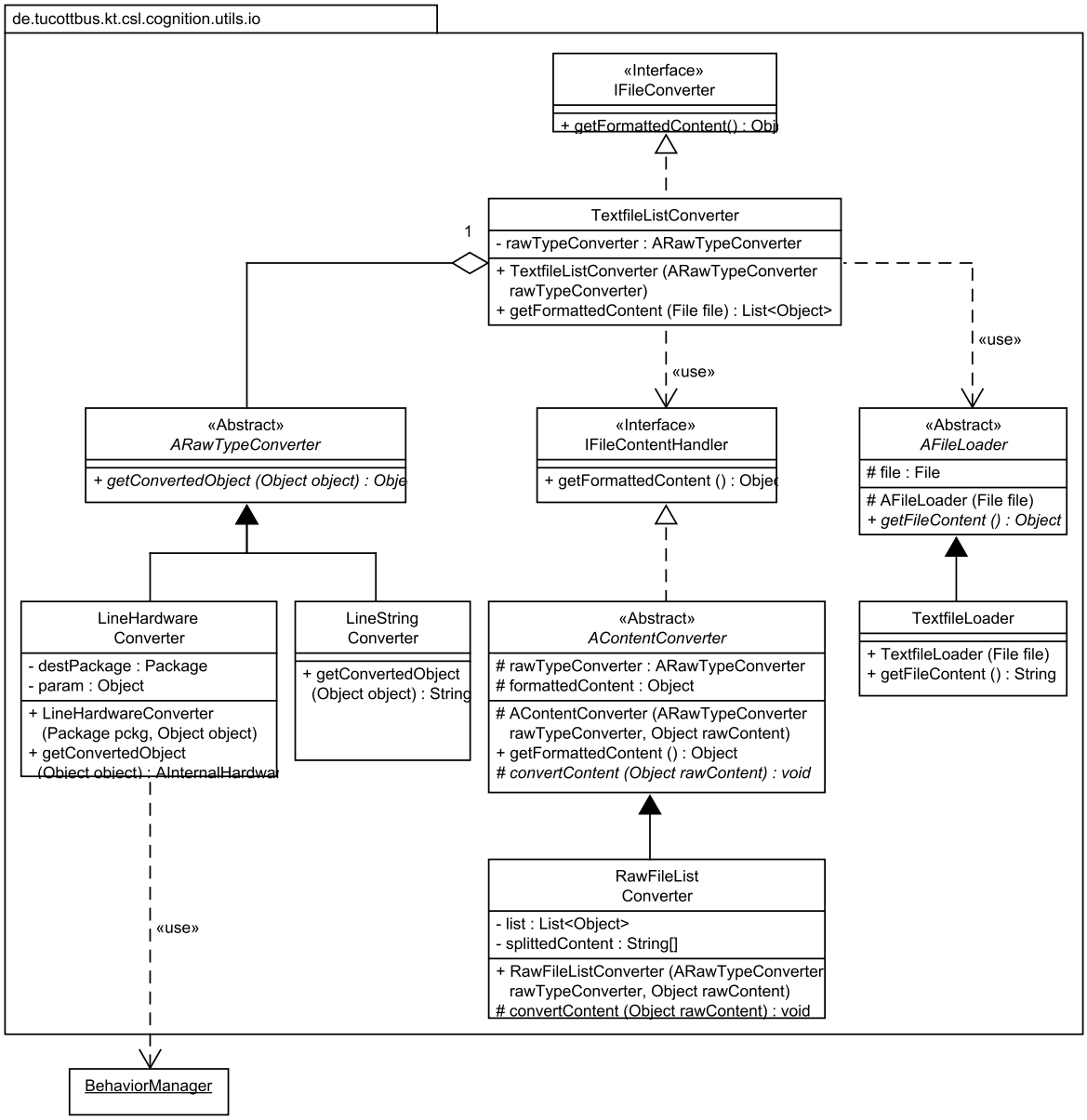


Abbildung 36: UML-Klassendiagramm zum Laden von verwendeten Modell Dateien.

Literaturverzeichnis

[1] *Cognitive System Lab room control software*. `https://github.com/matthias-wolff/CSL`. Zugriff: 2016-11-12.

[2] *Universal Cognitive User Interface (2015-2018)*. `https://www.researchgate.net/project/Universal-Cognitive-User-Interface-2015-2018`. Zugriff: 2016-11-10.

[3] Beat Pfister, T. K.: *Sprachverarbeitung*. Springer Berlin Heidelberg, 2008.

[4] Borthwick Andrew, Sterling John, A. E. G. R.: *Exploiting diverse knowledge sources via maximum entropy in named entity recognition*. In: *Proc. of the Sixth Workshop on Very Large Corpora*, Bd. 182, 1998.

[5] Grosz, Barbara J., S. C. L.: *Attention, Intentions, and the Structure of Discourse*. Comput. Linguist., 12(3):175–204, Juli 1986.

[6] Hamerich, S. W.: *Sprachbedienung im Automobil. Teilautomatisierte Entwicklung benutzerfreundlicher Dialogsysteme*. Springer, 1 Aufl., 2009.

[7] Haun, M.: *Cognitive Computing, Steigerung des systemischen Intelligenzprofils*. Springer Vieweg, 1 Aufl., 2014.

[8] Haykin, S.: *Cognitive Dynamic Systems*. Cambridge University Press, 2012.

[9] Hoffmann, R. und M. Wolff: *Intelligente Signalverarbeitung 2: Signalerkennung*. Springer Vieweg, 2 Aufl., 2015.

[10] Kephart, J. O. und J. Lenchner: *A Symbiotic Cognitive Computing Perspective on Autonomic Computing*. In: *Autonomic Computing (ICAC), 2015 IEEE International Conference on*, S. 109–114, July 2015.

[11] Klimczak, P., G. Wirsching und M. Wolff: *Kondome verschlimmern nur das Problem: Eine formale Semantik-Analyse des AIDS-Diskurses um Papst Benedikt*. Veröffentlichung in 2016 geplant.

[12] Lindemann, J.: *Interpretation und Artikulation mit Äußerungs-Bedeutungs-Transduktoren*. Doktorarbeit, Brandenburgische Technische Universität Cottbus-Senftenberg, 2016.

[13] Lindemann, J.: *Semantische Interpretation und Artikulation mit Äußerungs-Bedeutungs-Transduktoren*. In: Jokisch, O. (Hrsg.): *Tagungsband, 27. Konferenz Elektronische Sprachsignalverarbeitung, 02.-04.03.2016, Leipzig*, S. 119 – 126, 2016.

[14] Lison, P.: *Structured Probabilistic Modelling for Dialogue Management*. Doktorarbeit, University of Oslo, 2013.

[15] Markus Huber, Christian Kölbl, R. L. R. R. G. W.: *Semantische Dialogmodellierung mit gewichteten Merkmal-Werte-Relationen*. In: Hoffmann, R. (Hrsg.):

Tagungsband, 20. Konferenz Elektronische Sprachsignalverarbeitung 21.-23.09.2009, Dresden, S. 25 – 33, 2009.

[16] Martin, J. H. und D. Jurafsky: *Speech and language processing*. International Edition, 710, 2000.

[17] McTear, M. F.: *Spoken Dialogue Technology. Toward the Conversational User Interface.*. Springer, 1 Aufl., 2004.

[18] Ramachandran, D. und A. Ratnaparkhi: *Belief Tracking with Stacked Relational Trees*. In: *SIGDIAL Conference*, 2015.

[19] Ramachandran, D., P. Z. Yeh, W. Jarrold, B. Douglas, A. Ratnaparkhi, R. Provine, J. Mendel und A. Emfield: *An end-to-end dialog system for TV program discovery*. In: *Spoken Language Technology Workshop (SLT), 2014 IEEE*, S. 602 – 607, Dec 2014.

[20] Römer, R., W. Günther und M. Wolff: *Ein Beitrag zu den Natur- und Geisteswissenschaftlichen Grundlagen kognitiver Systeme*. In: Wagner, P. (Hrsg.): *Elektronische Sprachsignalverarbeitung 2013, Tagungsband, Bielefeld, 2013*, S. 93 – 102, 2014.

[21] Römer, R. und M. Wolff: *Konzeption eines Kognitiven Systems für den experimentellen Einsatz in Forschung und Lehre*. In: Wirsching, G. (Hrsg.): *Tagungsband, 26. Konferenz Elektronische Sprachsignalverarbeitung, 25.-27.03.2015, Eichstätt*, S. 212 – 223, 2015.

[22] Sarikaya, R.: *Personal Digital Assistants*. Keynote speaker INTERSPEECH 2015, September 2015. Dresden.

[23] Sheth, A., P. Anantharam und C. Henson: *Semantic, Cognitive, and Perceptual Computing: Paradigms That Shape Human Experience*. Computer, 49(3):64–72, Mar 2016.

[24] Starke, G.: *Effektive Softwarearchitekturen*. Hanser Verlag, 208.

[25] Traum, D. R. und S. Larsson: *The Information State Approach to Dialogue Management*, S. 325–353. Springer Netherlands, Dordrecht, 2003.

[26] Wirsching, G.: *Calculating semantic uncertainty*. In: *Proc. 2012 IEEE 3rd International Conference on Cognitive Infocommunications (CogInfoCom)*, S. 71–76, Dec 2012.

[27] Wirsching, G. und R. Lorenz: *Towards meaning-oriented language modeling*. In: *IEEE 4th International Conference on Cognitive Infocommunications (CogInfoCom), 2013 : 2 - 5 Dec. 2013, Budapest, Hungary ; proceedings*, S. 369–374. IEEE, Piscataway, NJ, 2013.

[28] Wirsching, G. und M. Wolff: *Semantische Dekodierung von Sprachsignalen am Beispiel einer Mikrofonfeldsteuerung*. In: Hoffmann, R. (Hrsg.): *Elektronische Sprachsignalverarbeitung 2014, Tagungsband der 25. Konferenz Dresden, 26. – 28. März 2014*, S. 104 – 109, 2014.

[29] Wolff, M.: *Akustische Mustererkennung*. TUDpress, 2011.

[30] Wolff, M., R. Römer und G. Wirsching: *Towards coping and imagination for cognitive agents*. In: *6th IEEE International Conference on Cognitive Infocommunications (CogInfoCom) Györ (Hungary), October 9-21, 2015*, S. 307 – 312, 2016.

[31] Young, S.: *Cognitive User Interfaces*. IEEE Signal Processing Magazine, 27(3):128–140, May 2010.

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

[33] Young, S. J.: *Still Talking to Machines*. In: *INTERSPEECH*, 2010.

[34] Zimmerli, W. C.: *Künstliche Intelligenz. Die Herausforderung der Philosophie durch den Computer*. Forum Interdisziplinäre Forschung, Januar 1988.