T U M

I N S T I T U T F Ü R I N F O R M A T I K

Aufgabenbeschreibung mit verhaltensbasierter Robotersteuerung und natürlicher Kommunikation

Markus Rickert, Michael Kaßecker, Alois Knoll

TUM-I0914
Juni 09

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-06-I0914-0/1.-FI

Druck: Institut für Informatik der
Technischen Universität München

Aufgabenbeschreibung mit verhaltensbasierter Robotersteuerung und natürlicher Kommunikation*

Markus Rickert, Michael Kaßecker, und Alois Knoll

Lehrstuhl für Echtzeitsysteme und Robotik
Institut für Informatik
Technische Universität München

Zusammenfassung In diesem Bericht wird ein System zur Mensch-Roboter-Kommunikation zum Zwecke der abstrakten Aufgabenbeschreibung vorgeschlagen. Bei der herkömmlichen Programmierung werden Fahrtenprogramme üblicherweise individuell für einzelne Modelle, Aufgaben und Umgebungen erstellt. Bei Änderungen der Rahmenbedingungen müssen Quelltexte in der Regel aufwendig angepasst werden. Im Gegensatz dazu werden Aufgabenbeschreibungen in diesem hier vorgeschlagenen System auf abstrakterer, höherer Ebene angestrebt. Während bei den bekannten Programmierparadigmen Expertenwissen und Rahmenbedingungen implizit und daher nicht wiederverwendbar im Quelltext eingebettet sind, wird bei dem in dieser Arbeit vorgeschlagenen System solches Wissen explizit angegeben. Dies führt zu einer aufgabenorientierten Beschreibung, bei der selbst solche Rahmenbedingungen automatisiert berücksichtigt werden, wenn sie wie in der natürlichen Mensch-Mensch-Kommunikation als selbstverständlich vorausgesetzt werden.

1 Einleitung

Der moderne industrielle Fertigungsprozess ist durch einen hohen Automatisierungsgrad und immer kürzere Produktzyklen gekennzeichnet. Insbesondere gilt dies für die Massenfertigung, bei der Teile in hoher Stückzahl mit möglichst gleicher Qualität hergestellt werden. Darüber hinaus zeichnet sich in den letzten Jahren ein Trend zur individualisierten Produktion, bei der auf Kundenwunsch spezielle Varianten der Produkte in variabler Stückzahl gefertigt werden müssen, ab. Dem gegenüber stehen verhältnismäßig hohe Anschaffungskosten, welche erwartungsgemäß für aufgabenspezifische Maschinen und Werkzeuge zu veranschlagen sind. In finanziell-investitorischer Hinsicht erscheint es deswegen geboten, bei Aufgaben mit hoher Spezifität und Variabilität bevorzugt solche Roboter und Fertigungsanlagen zu verwenden, die sich flexibel einsetzen lassen und somit einen höheren Wiederverwendungsgrad bei schneller Anpassbarkeit

* Diese Arbeiten werden von der Deutschen Forschungsgesellschaft (DFG) im Rahmen des Exzellenzclusters „CoTeSys“ (www.cotesys.org) gefördert.

an neue Fertigungsaufgaben garantieren können. Durch die angesprochene Maßnahme lassen sich etwa Umrüstzeiten effektiv verkürzen und eine lukrative Fertigung in Kleinserie ermöglichen. Im Folgenden wird ein Ansatz zur generellen, schnelleren, intuitiven Roboter-Mensch Kommunikation in der Aufgabeschreibung vorgeschlagen.

2 Motivation

Üblicherweise wird ein Roboter in einem sogenannten *Teach-In*-Verfahren an eine neue Aufgabe angepasst. Dies bedeutet genaues und explizites Definieren sequenzieller Handlungsfolgen (*Scripting*). Neben der naheliegenden Möglichkeit, Positionen sowie Posen oder allgemein Aktionen und Aktivitäten zum Beispiel unmittelbar als Software bzw. als *Script* anzugeben[1], wird ein Programm alternativ – sozusagen als erster Schritt in Richtung intuitiver Bedienung – durch Anfahren einer Position mit Hilfe einer Steuerkonsole eingegeben. In beiden Fällen können die verfügbaren Ein- und Ausgangssignale zur Synchronisation der unterschiedlichen Abläufe herangezogen werden. Um die dem Fertigungsprozess innewohnenden Unwägbarkeiten in gewissem Umfang zu begegnen, können Kraftsensoren zum Ausgleich von Ungenauigkeiten verbaut werden.

Hier sei darauf hingewiesen, dass die Anpassung speziell für genau *einen* Roboter und für *genau eine* Aufgabe durchgeführt wird. Offensichtlich ist eine Wiederverwendung in anderem Zusammenhang – beispielsweise mit verändertem Roboter oder in unterschiedlicher Umgebung – nicht zu garantieren. Sogar unter der Voraussetzung des selben *Robotertyps* – etwa einem sechsachsigen Industrieroboter – jedoch ersetzt durch eine sich leicht in der Form, Dimension oder gar Position unterscheidende Variante, ist die ordnungsgemäße Erfüllung der ursprünglich intendierten Aufgabe in vielen Fällen nicht mehr zu erwarten. Auch die konsequente Verwendung eines gemeinsamen Weltkoordinatensystems, bei der alle Positionen im kartesischen Raum spezifiziert werden, vermag hier nur bedingt Abhilfe zu schaffen, da hierdurch weder Aufgaben noch einzelne Aktionen oder die erfüllte Funktion als solche explizit definiert sind. Wird ein Manipulator ausgetauscht oder versetzt und dabei das Referenzsystem in Verbindung mit der Kinematik angepasst, besteht während der Fahrt immer noch die Möglichkeit von Kollisionen mit der Umgebung oder anderen Manipulatoren. Bisher wurde dieses Problem durch individuelle Angabe von Ausweichfahrten gelöst. Es wäre wünschenswert, diesen Schritt zukünftig automatisiert erfolgen zu lassen. Somit soll hier unmittelbar die Forderung abgeleitet werden, Aufgaben und Aktionen in Zukunft Roboter- und umgebungsunabhängig angeben zu können.

Herkömmliche Quelltexte weisen im Hinblick auf die Wiederverwendbarkeit eine weitere Schwachstelle auf. Als explizite Anweisungsbeschreibung auf niederer Ebene ist das Entwicklerwissen *implizit* enthalten. Weder *Bedeutung* noch *Funktion* bzw. *Zweck* der Aktionen sind im Sinne von *„Setze Schraube 22a in*

[1] Hierin eingeschlossen sind selbstverständlich auch graphisch gestützte Eingabgemöglichkeiten über eine GUI wie sie in kommerziellen Produkten – etwa MelfaWorks oder RobotWorks – zu finden sind.

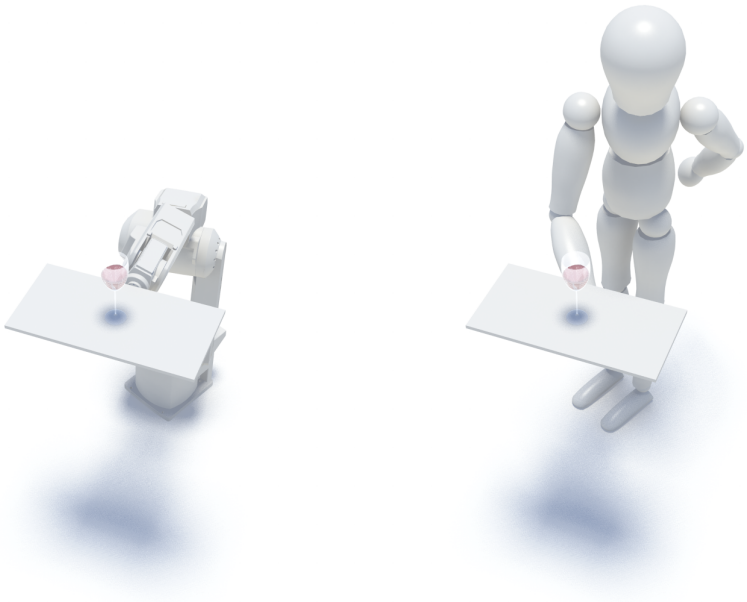

Abbildung 1. Transport eines Weinglases auf einem Tablett. Interpretation der abstrakten Aufgabenbeschreibung mit einem Industrieroboter und einem Humanoiden.

das obere Bohrloch B07“ im herkömmlichen Quelltexten wiederverwendbar kodiert, noch sind die essentiellen Rahmenbedingungen unmittelbar und eindeutig aus dem Quelltext zu entnehmen. Über den Umstand, dass es sich bei der Schraube um ein Rechtsgewinde handelt und somit ausschließlich in entsprechenden Bohrlöcher geschraubt werden darf, schweigt sich der Quelltext beharrlich aus. Dies erschwert es einer am ursprünglichen Entwicklungsprozess unbeteiligten Person, welche ein Programm nach einigen Monaten erweitern oder modifizieren soll, die Intention des Entwicklers ohne einen ausführlichen Kommentar selbstständig zu erschließen. Werden Anpassungen in der Architektur der Fabrik oder Fertigungsstraße vorgenommen und bisher nicht verwendete Robotertypen eingesetzt, muss die Software selbst dann neu entwickelt werden, wenn sich das Produkt nicht verändert hat. Unter Umständen muss notwendiges Wissen über den Fertigungsprozess sowie erwähntes *Hintergrundwissen* vom Entwickler neu bzw. erneut erworben werden. Ähnliches kann auch auf Modifikationen zutreffen, wie sie bei der Herstellung von kleinen, individuellen Serien notwendig sind. Die Komplexität steigt hierbei insbesondere bei der Wandlung von klassischen Roboterarchitekturen in flexiblere, modular aufgebaute Konzepte. Vorteile bei der Anpassung von Robotern an eine Aufgabe könnten sich daraus ergeben, dass solches domänenspezifisches *Hintergrundwissen* zum Teil problem- und aufgabenunabhängig (*statisch*) ist und sich unabhängig von der Aufgabe etwa in Ontologien bzw. Datenbanken organisieren und verwalten lässt. Der Entwickler

zukünftiger Systeme sieht sich also an dieser Stelle verstärkt mit Fragen der Wissensorganisation und seiner Anwendung konfrontiert.

Ein weiteres Problem beim Programmieren von Anweisungen besteht darin, dass lediglich ausgebildete Fachkräfte in der Lage sind neue Fahrten bzw. Aktionen vorzugeben, da zum einen fortgeschrittene Kenntnisse in Robotik und zum anderen ein sicherer Umgang mit den Fachtermini vorausgesetzt werden. Die aus Benutzersicht eindeutige Anweisung *„die Tasse dort drüben aufzunehmen“* genügt zur Definition einer Roboteranweisung *noch* nicht. Gegenwärtig muss eine solche Anweisung vom Programmierer zunächst in Teilschritte zerlegt und anschließend in Steuerungsanweisungen transformiert werden. Außerdem ist eine anspruchsvollere Integration von Sensordaten vorzunehmen und räumliche Zusammenhänge und ihre Abhängigkeiten zu berücksichtigen. Weitere Nebenbedingungen, wie das Vermeiden von Kollisionen mit der Umgebung oder des Roboterarms mit sich selbst, lassen sich zum gegenwärtigen Zeitpunkt nur durch Intuition des Programmierers umsetzen. Eine echte Herausforderung besteht allerdings darin, dass in der natürlichsprachlichen Anweisung bestimmte Informationen als bekannt vorausgesetzt werden. Einer erwachsenen Person erscheint es intuitiv selbstverständlich, dass der Inhalt einer aufgenommen Tasse beim anschließenden Transport nicht verschüttet werden darf. Wenn es einem Robotersystem ermöglicht werden soll, eine aus menschlicher Sicht möglichst intuitiv beschriebene Aufgabe *(vgl. natürlichsprachlich)* zu verstehen und gleichzeitig eine solche Beschreibung unabhängig von einem speziellen Robotertyp gültig sein soll, so besteht folglich eine besondere Herausforderung darin, solche Nebenbedingungen automatisch zu berücksichtigen.

3 Verhaltensbasierte Robotersteuerung

In [1] wurde erstmals ein Konzept vorgeschlagen, um verschiedene Bedingungen während der Ansteuerung eines Roboters miteinander zu verknüpfen. Im Besonderen wird die gewohnte Vorgehensweise, nämlich einzelne Robotergelenke unmittelbar anzusteuern, durch geschickte Definition von Endeffektor bezogenen Anforderungen hinsichtlich des Dynamik- und des Kräfteverhaltens als *Aufgaben* (*Tasks*), ersetzt. Darüber hinaus können *Nebenbedingungen*, wie etwa die Vermeidung von Singularitäten oder unterschiedliche Prioritäten bei der Aufgabenerfüllung angegeben werden, welche unter Verwendung redundant ausgelegter Manipulatoren berücksichtigt werden. Auf diese Weise kann das Ausweichen von Hindernissen ebenso wie die Kontrolle mehrerer Endeffektoren elegant integriert werden. Exemplarisch für eine solch komfortable Formulierung von parallelen Anforderungen kann etwa ein humanoider Roboter angeführt werden, welcher das Gleichgewicht hält, während er mit der rechten Hand eine Kraft auf eine Oberfläche ausübt und dieser Bewegung mit dem Kopf folgt. Eine Kombination mehrerer Bedingungen kann über Prioritäten zu einer einzelnen Aufgaben zusammengefasst werden, die wiederum über die Verbindung von festgelegten Ereignissen zu einem Verhalten kombiniert werden können. Die Fortbewegung eines Humanoiden kann so als Sequenz angegeben werden [2,3]. Eine derarti-

ge Abfolge könnte etwa mit folgenden Einzelanweisungen beschrieben werden: Zunächst das *Gewicht auf linken Fuß zu verlagern*, danach den *rechten Fuß nach vorne zu bewegen*, und anschließend das *Gewicht auf den rechten Fuß zu verschieben* und abschließend den *linken Fuß nach vorne zu bewegen.*

Die Beschreibung der dynamischen Zusammenhänge im kartesischen Raum kann aus der dynamischen Beschreibung im Gelenkraum hergeleitet werden. Die aufzubringenden Drehmomente τ für eine vorgegebene Beschleunigung $\ddot{\theta}$ in Position θ mit Geschwindigkeit $\dot{\theta}$ lassen sich dort aus der Massenmatrix M, den Zentrifugal- und Korioliskräften V und dem Gravitationsvektor G bestimmen:

$$\tau = M(\theta)\ddot{\theta} + V(\theta, \dot{\theta}) + G(\theta).$$

Ein entsprechender Zusammenhang für eine Kraft F würde sich dementsprechend für eine kartesische Beschleunigung $\ddot{x}$ mit den entsprechenden Komponenten im kartesischen Raum M_x, V_x und G_x darstellen als:

$$F = M_x(\theta)\ddot{x} + V_x(\theta, \dot{\theta}) + G_x(\theta).$$

Über die Verbindung zwischen Kräften und Drehmomenten, sowie zwischen Geschwindigkeiten in den beiden Räumen durch die Jacobi-Matrix J

$$\begin{aligned}\tau &= J^T(\theta)F \\ \dot{x} &= J(\theta)\dot{\theta}\end{aligned}$$

und der Ableitung

$$\ddot{x} = \dot{J}(\theta, \dot{\theta})\dot{\theta} + J(\theta)\ddot{\theta}$$

ergibt sich damit ein Zusammenhang von Drehmomenten im Gelenkraum und einer Dynamikbeschreibung im kartesischen Raum:

$$\tau = J^T(\theta)\left(M_x(\theta)\ddot{x} + V_x(\theta, \dot{\theta}) + G_x(\theta)\right)$$

mit

$$\begin{aligned}M_x(\theta) &= J^{-T}(\theta)M(\theta)J^{-1}(\theta) \\ V_x(\theta, \dot{\theta}) &= J^{-T}(\theta)\left(V(\theta, \dot{\theta}) - M(\theta)J^{-1}(\theta)\dot{J}(\theta, \dot{\theta})\dot{\theta}\right) \\ G_x(\theta) &= J^{-T}(\theta)G(\theta).\end{aligned}$$

Hierbei laßt sich auch eine dynamisch konsistente, generalisierte Inverse der Jacobi-Matrix J ableiten:

$$\begin{aligned}J^{-1}(\theta) &= M^{-1}(\theta)J^T(\theta)M_x(\theta) \\ M_x(\theta) &= \left(J(\theta)M^{-1}(\theta)J^T(\theta)\right)^{-1} \\ J^{-1}(\theta) &= M^{-1}(\theta)J^T(\theta)\left(J(\theta)M^{-1}(\theta)J^T(\theta)\right)^{-1}.\end{aligned}$$

Da bei hochredundanten Manipulatoren mehr Gelenke zur Verfügung stehen als für die Einhaltung von Bedingungen auf den Endeffektor nötig sind, lassen

sich eine Vielzahl von Gelenkbewegungen ausführen, ohne dabei den Endeffektor zu beeinflussen. Diese Menge an Bewegungen lässt sich über eine Projektion in den entsprechenden dynamisch konsistenten *Nullspace* N_{Task} sowohl für Drehmomente, als auch Geschwindigkeiten im Gelenkraum abbilden:

$$\begin{aligned} \dot{x} &= J(\theta)\dot{\theta} & F &= J^{-T}(\theta)\tau \\ \dot{q} &= J^{-1}(\theta)\dot{\theta} & \tau &= J^{T}(\theta)F \\ N(\theta)_{\text{Task}} &= I - J^{-1}(\theta)J(\theta) & N^{T}(\theta)_{\text{Task}} &= I - J^{T}(\theta)J^{-T}(\theta). \end{aligned}$$

Eine den Endeffektor betreffende Aufgabe kann auf diese Weise eingehalten werden, während dennoch versucht wird nebenbei eine gewünschte Pose einzuhalten:

$$\begin{aligned} \tau &= \tau_{\text{Task}} + N^{T}_{\text{Task}}(\theta)\tau_{\text{Posture}} \\ \dot{q} &= \dot{q}_{\text{Task}} + N_{\text{Task}}(\theta)\dot{q}_{\text{Posture}}. \end{aligned}$$

Mehrere parallel auszuführende Aufgaben können nun mit Prioritäten versehen und miteinander kombiniert werden durch eine Projektion in den jeweiligen *Nullspace* der höherliegenden Aufgabe [4]:

$$\begin{aligned} \tau &= \tau_1 + N_1^T(\theta)\left(\tau_2 + N_2^T(\theta)\left(\tau_3 + \ldots\right)\right) \\ \dot{q} &= \dot{q}_1 + N_1(\theta)\left(\dot{q}_2 + N_2(\theta)\left(\dot{q}_3 + \ldots\right)\right). \end{aligned}$$

4 Beschreibung des Verfahrens und verwendete Methoden

Bei Betrachtung des Informationsflusses (beginnend mit der natürlichen Sprache bis hin zur ausgeführten Handlung bzw. Manipulation) wird deutlich, dass Entscheidungen auf verschiedenen Abstraktionsebenen zu treffen sind. Im Groben ergeben sich zunächst drei größere Einheiten. Auf die Lokalisierung, die Beseitigung von Widersprüchen, die Inferenz oder ähnliches sind in diesem Bericht nicht in der gebotenen Tiefe eingegangen – sie werden in eigenen Modulen zu einem späteren Zeitpunkt realisiert und in hierauf folgenden Arbeiten Gegenstand intensiverer Betrachtungen sein. In Unterabschnitt 4.1 wird auf diese Thematik und die Anforderungen an die Wissensbasis oberflächlich eingegangen. Auf der ersten Ebene soll der natürliche Ausdruck, d.h. Sprache und Gestik in eine reguläre Form überführt werden. In der unterliegenden Schicht sind – etwa durch Datenbankabfrage – die Bedingungen für die Aufgabe zu ermitteln. Darunter befindet sich eine roboterspezifische Schicht, welche die Fahrbefehle umsetzt. Auf dieser Ebene sind Bedingungen und Prioritäten gekapselt. Es ist zu betonen, dass erwähnte Schichten nach Möglichkeit voneinander unabhängig zu konzipieren sind und daher definierte Schnittstellen für den Einsatz mit mehreren, austauschbaren Robotern unabdingbar sind. Hierfür ist eine spezielle Methode zur Ansteuerung notwendig (siehe Abschnitt 3). In Abbildung 2 sind die Schichten kurz skizziert. In Abschnitt 4.2 ist der Schritt vom natürlichen Ausdruck zum Aktionsprimitiv beschrieben. Unterabschnitt 4.3 behandelt die Aktivitätsschicht und umfasst ebenfalls die Abläufe in der Ausführungsschicht.

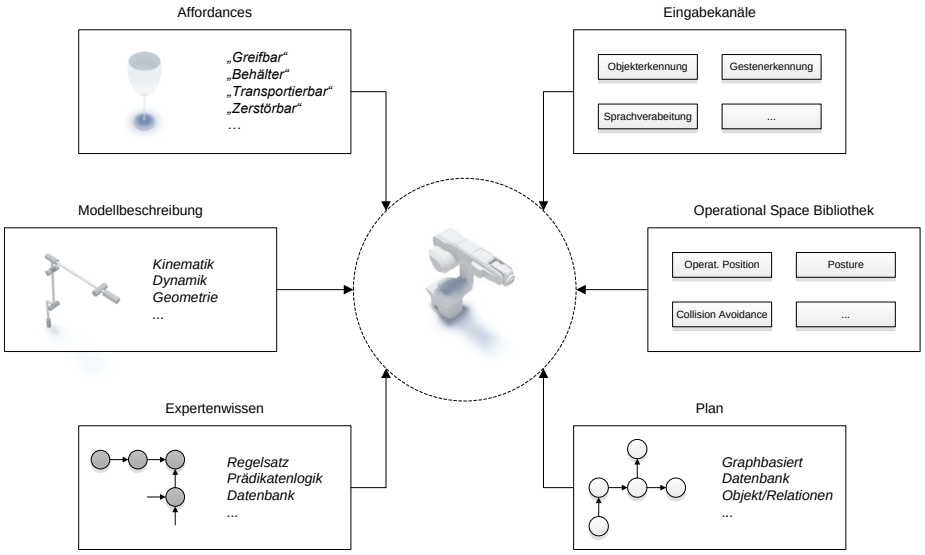

Abbildung 2. Blick auf das System und Darstellung der einzelnen Module. Eine Aufgabenbeschreibung in natürlicher Sprache resultiert in abstrakter Form (Plan) unter Einbezug von Hintergrund- und Expertenwissen. Das konkrete Programm für ein spezielles Robotermodell wird ferner unter Berücksichtigung der Modellbeschreibung und der *Operational Space*-Bibliothek generiert.

4.1 Wissenstrukturen und ihre Organisation

Wie bereits in Abschnitt 2 erwähnt wird angenommen, dass bestimmte Sachverhalte – über einen gewissen Zeitraum hinweg – statisch und Aufgabenunabhängig ausgedrückt werden können. Beispielsweise hat eine Schraube X stets ein Rechtsgewinde, ein Zahnrad Y einen festen Axialmodul oder Wasser die Eigenschaft bei Raumtemperatur flüssig zu sein. Neben der Eigenschaftsbeschreibung von Objekten – in Verbindung mit einer oder mehrerer Bedingungen unter der die Eigenschaft Gültigkeit besitzt – umfassen solche Beschreibungen zusätzlich noch den *Angebotscharakter* [5,6] von solchen Objekten. Beispielsweise ist hierin kodiert, unter welchen Bedingungen eine Handlung mit einem Objekt durchgeführt werden kann. So ist ein Ball nur dann von speziellen Robotern greifbar, wenn er nicht rollt, eine Kerze nur dann gefahrlos transportierbar, wenn sie nicht brennt. Der *Angebotscharakter* wohnt Objekten insofern inne, als dass sie unterschiedliche Möglichkeiten ihres Gebrauchs bieten. Ein Wasserglas macht in diesem Paradigma das Angebot durch einen bestimmten Typ Greifer aufgenommen, oder etwa ein Hammer von diesem *ergriffen zu werden*[2]. Hier besteht im Übrigen eine Nähe zu agentenbasierten Systemen, bei denen Agenten einen bestimmten *Dienst anbieten*. Eine Erweiterung in diesem Sinne wird ggf. erwogen. Neben ei-

[2] Man muss hier unterscheiden zwischen der Repräsentation und der Bedeutung. Auch das Wort kann *ergriffen werden*

ner Eigenschaftsbeschreibung kann zudem Struktur und Zusammensetzung von Objekten in einem anderen Modell ausgedrückt werden. In dieser Arbeit wird jedes manipulierbare Objekt in diesem Sinne betrachtet und strukturiert.

Hiervon ist *prozedurales* Wissen zu unterscheiden. Unter prozeduralem Wissen werden Handlungsfolgen verstanden, die einem definierten Ziel dienen. Ein Kochrezept etwa ist hierfür ein einfaches Beispiel. Es wird angenommen, dass sich solche Strukturen und Beziehungen getrennt speichern und verwalten lassen. Intuitiv ist einzusehen, dass Anpassungen auf diese Weise schneller vorzunehmen sind, allerdings sind auch Fälle denkbar in denen sich Struktur und prozedurales Wissen im Sinne einer Implikation bedingen. Ändert sich beispielsweise der Bauplan, so ändert sich auch der Produktionsvorgang. Ziel muss es also auch sein, Änderungen – in begrenztem Umfang – in der Wissensbasis automatisch vornehmen zu können und Verhalten anzupassen (*vgl. Inferenz).* Hierfür ist ein spezielles Form der Kodierung der Prozeduren notwendig (*vgl. Mittel-Zweck-Beziehung*). Folgendes Beispiel soll verdeutlichen, wie eine Roboter unabhängige Aufgabenbeschreibung realisiert werden könnte. Die Aufgabe bestehe darin, ein gefülltes Wasserglas zu transportieren. Üblicherweise wird ein Mensch keine solchen detaillierten Anweisungen geben, die für eine Programmierung eines Roboters genügen, sondern einfach verlangen, ein Glas vom Tisch aufzunehmen und es auf einer anderen Ablage abzusetzen. Die Zerlegung in entsprechende Teilaufgaben muss hierbei nun selbständig erfolgen. Eine reguläre Beschreibung der Aufgabe könnte die Bedingung enthalten, dass das Wasserglas nicht verschüttet werden darf. Die *allgemeine Bedeutung* von *nicht Verschütten* lässt sich zum Beispiel als Bedingung in Abhängigkeit von Transportwinkel, Geschwindigkeit und Ruck ausdrücken. Die darunter liegenden Fahrbefehle sind in Roboterprogrammen noch viel detaillierter hinterlegt.

4.2 Vom natürlichen Ausdruck zum Aktionsprimitiv

Der Prozess der Überführung von natürlicher Sprache in ein Aktionsprimitiv erfolgt in zwei größeren Schritten. Über ein herkömmliches Spracherkennungswerkzeug wird in einem ersten Schritt natürliche Sprache in eine Textform überführt, während im nachfolgenden eindeutige, regelbasierte Anweisungen für die Aktivitätenebene generiert werden. Analoges gilt für das Verwenden von Gesten mit dem Unterschied, dass hier von einem Gestenerkenner mit einem Zeitstempel versehene *Events* an die nächste Stufe übergeben werden.

Nach Erzeugung der textuellen Eingaben erfolgt eine syntaktische und semantische Analyse. Hierbei wird die Texteingabe zunächst daraufhin überprüft, ob sich der Text lediglich aus bekannten Wörtern zusammensetzt. Danach erfolgt die Identifikation von Phrasen. Bei unbekannten Wörtern oder Wortgruppen wird gegebenenfalls anhand einer Datenbank geprüft, ob es ähnliche Worte oder Satzteile gibt, die bei Ersetzung einen bekannten, sinnvollen Zusammenhang bilden und gegebenenfalls durch diese ersetzt. Des weiteren sind Worte und Wortgruppen in einer Datenbank als Relation von Wörtern und Kategorie angelegt, anhand derer festgestellt werden kann, ob ein Wort oder Wortgruppe in einem korrekten grammatischen Zusammenhang steht.

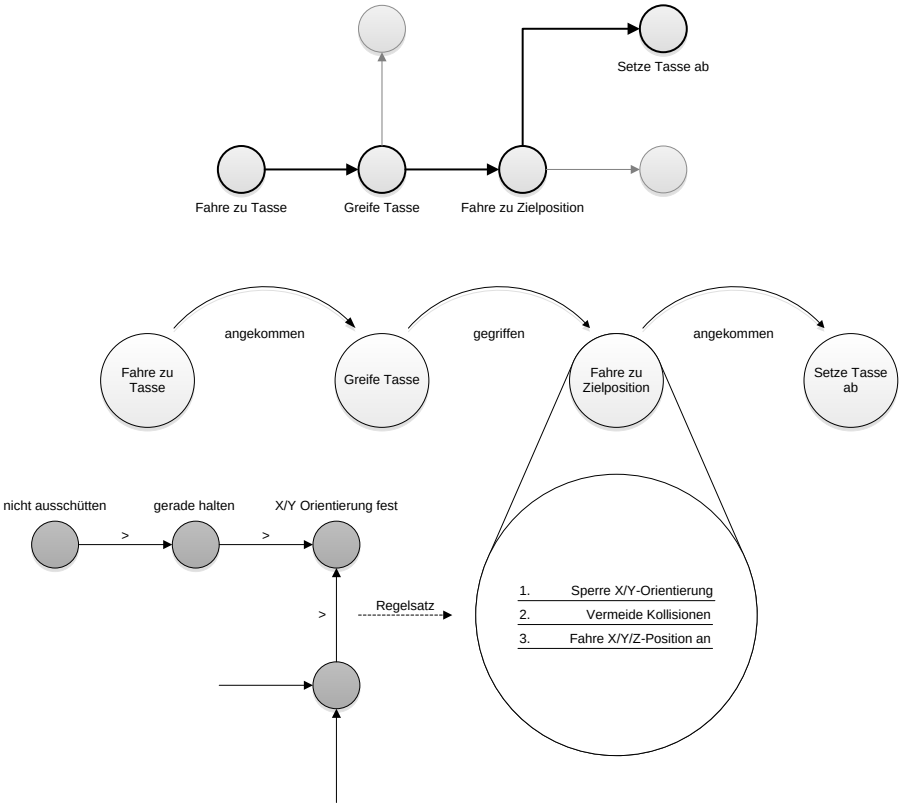

Abbildung 3. Beispielhafte Verhaltens- und Aktionsbeschreibung. Identifizierung der Einzelaktion mittels eines Inferenzalgorithmus (oben). Darstellung des Aktionsplans (mitte) und Ermittlung der Prioritäten (unten).

Ausgewählte Wörter oder Phrasen sind in der Datenbank als *Trigger* hinterlegt. Es werden solche Wörter und Phrasen in die Datenbank eingetragen, die Menschen typischerweise in der Mensch-Roboter-Interaktion nutzen. Hierfür können - ausgehend von den Fähigkeiten des Roboters – typische Äußerungen experimentell ermittelt werden. Mit jedem *Trigger* wird eine oder mehrere hinterlegte grammatische Strukturen assoziiert, die wiederum in einer *regelbasierten Instruktion* resultierten. Allgemein ausgedrückt werden unter *Instruktionen* Anweisungen mit Seiteneffekten verstanden, anhand derer Objekte instantiiert und Objektbeziehungen manipuliert werden können. Eine solche Instruktion besteht aus einer Anweisung und einer in Anzahl und Typ variablen Menge von Parametern. Die Interpretation einer solchen Instruktion ist adressatenabhängig und wird an dieser Stelle nicht weiter für einen allgemeinen Fall spezifiziert.

Im Speziellen werden diese Instruktionen eingesetzt, um etwa die Reihenfolge der Prioritäten zu ändern. In einem einfachen Beispiel wie *Put the glass of water on the table.* wird zunächst eine syntaktische Analyse durchgeführt. In der Daten-

bank ist das Verb *put* an mehreren Stellen hinterlegt. Zum einen befindet sich es sich in der Wörterliste, zum anderen seien mehrere Einträge als *Trigger* vorhanden. Der passende Eintrag bestehe etwa in `Put <object> <desc> <place>` verknüpft mit einer Information, welche Instruktionssequenzen ausgeführt werden sollen. Unter anderem werden solche Instruktionen ausgeführt, die neben der Anweisung, das Objekt „Glas gefüllt mit Wasser“ zu identifizieren und zunächst einmal im Speicher zu generieren, sowie die Ortsbeschreibung auflöst. Danach wird ein Aktionsplan erstellt. Dieser beinhaltet eine Beschreibung von atomaren Aktionen, wie etwa an das Glas heranzufahren, es zu ergreifen, es anzuheben, es zu transportieren und dann wieder abzusetzen.

Ein Aktionsplan wird wie folgt erstellt: Die Datenbank enthält die Informationen über durchführbare Aktivitäten als *Ziel-Bedingungs-Relationen. Ziele* werden als hierarchische Beziehung von Ober- und Unterzielen abgelegt [7,8]. Ähnlich wie in einem State-Chart hat jedes Ziel keine, eine oder mehrere Vor- und Nachbedingungen. Anhand dieser Bedingungen lässt sich ein Graph erstellen. Sofern ein Oberziel von einem Ausgangszustand erreichbar ist, lässt sich ein Pfad durch diesen Graph finden. Umgekehrt ist es so, dass für den nicht erreichbare Oberziele ein solcher Pfad trivialerweise nicht existiert. In einem solchen Fall kann ein Dialogmanager zur Problemlösung eingesetzt werden, indem das System beispielsweise explizit nachfragt, wie es ein Ziel erreichen soll oder kann.

Es sei vorausgesetzt, dass die Aktionssequenz nun ermittelt bekannt sei. Als nächster Schritt sind Verhaltensbeschreibungen für den *Operational Space* zu ermitteln. In der Datenbank ist grundlegendes Wissen (z.B. *Axiome* oder *Symbole*) über die Priorisierung des Verhaltens als Regelsatz eines Logikkalküls enthalten. Es wird angenommen, dass die Prädikatenlogik für die hier betrachteten Anwendungsfälle ausreichend ist, denn die Aufgabe, die hier lediglich erfüllt werden soll besteht darin, bei Vorliegen der Fakten Implikationsregeln anzuwenden um eine geordnete Menge von Verhaltensindikatoren zu ermitteln. Nach Anwendung entsprechender Transformationsregeln ergibt sich als Schluss entweder eine gültige Prioritätenliste des Verhaltens für jede einzelne Aktion oder es kann kein Ergebnis ermittelt werden. Im zuletzt genannten Fall wird der Dialogmanager eine Fehlermeldung an den Nutzer zurückgeben.

Im Falle des Beispiels des *mit Wasser gefüllten Glases* ist aufgeführt, dass ein *mit Wasser gefülltes Glas* den Zustand *transportiert* einnimmt, wenn es zuvor an den *gewünschten Ort* abgesetzt wurde, d.h. zuvor die Aktion *absetzen* durchgeführt wurde. Das *Absetzen* wiederum sei erst dann möglich, wenn der Roboterarm an den *gewünschten Ort* gefahren ist, usw. Neben diesen Implikationen können Handlungsalternativen mit Priorisierung angegeben werden.

Der fertige Aktivitätsplan wird anschließend an die darunter liegende Aktivitätsebene übergeben.

4.3 Von der Aktivitäten- zur Ausführungsebene

Eine Aktion A kann als Komposition von Aufgaben (*Tasks*) T formuliert werden. Auf der Menge solcher Aufgaben ist bezüglich ihrer Priorität eine Ordnung definiert. Aufgaben erfüllen die Funktion aus bekannten Eingangsgrößen –

abhängig von der Robotersteuerung – wahlweise Drehmoment oder Geschwindigkeit auszugeben, d.h. Aufgaben kapseln Berechnungslogiken der Zielgrößen. Typische Eingabewerte sind etwa Zielposition, Orientierung oder ein Auswahlvektor, der die relevanten Freiheitsgrade bestimmt. Als weitere kanonische Eingabegrößen ist das Kinematik- und Dynamikmodell des Roboters zu nennen [9]. Aus diesen Eingangsgrößen werden nun die Zielgrößen (vgl. Drehmoment oder Geschwindigkeit) ermittelt. Typische Taskdefinitionen sind *Operational Position*, *Singularity Avoidance*, *Collision Avoidance* oder *Posture*. Diese wurden zuvor in einer Bibliothek implementiert und bereitgestellt. Ebenfalls in der Aufgabenlogik enthalten sind sogenannte *Projektionsregeln* $\triangleleft$. Diese werden benötigt, um Aufgaben niederer Priorität an höher priore unter der Voraussetzung anzupassen, dass Nebenbedingungen der höherwertigen Aufgabe eingehalten werden. Die Projektionsregeln entsprechen denen in Kapitel 3 beschriebenen Konzepten von *Nullspaces*.

Die im obigen Beispiel (siehe Abbildung 3) beschriebene Aktion kann in angesprochener Notation wie folgt dargestellt werden:

$$A_{\text{Fahre zu Zielpos.}} = T_{\text{Sperre X/Y-Orient.}} \triangleleft T_{\text{Vermeide Kollisionen}} \triangleleft T_{\text{Fahre X/Y/Z-Pos. an}}.$$

Die Robotersteuerung stellt nun sicher, dass im Taktschritt der Regelung die Eingabewerte der Aufgaben erfasst werden, d.h. die entsprechenden Sensoren müssen zunächst einmal ausgelesen werden. Für den Fall, dass eine Weiterverarbeitung der Werte notwendig sein sollte, müssen die entsprechenden Berechnungen durchgeführt werden. So müssen für die aktuelle Stellung des Manipulators die Dynamikberechnungen durchgeführt und die von den beteiligten Aufgaben gekapselten Gleichungssysteme gelöst werden. Aus dieser Berechnung resultiert entweder ein Geschwindigkeitsvektor oder ein Drehmoment, der an die Motorsteuerung übergeben wird.

Auf diese Weise ist die einfache Realisierung eines verteilten Systems möglich, bei der die komplexe Bildverarbeitung auf externe Rechner ausgelagert ist. Am Beispiel eines Handgestenerkenners kann der Vorteil einer solchen Vorverarbeitung deutlich gemacht werden. Anstelle von umfangreichen Bildrohdaten fällt hier lediglich ein Datenvolumen in Höhe eines sechsdimensionalen Zielgrößenvektors als Eingabe für eine *Operational Position*-Aufgabe an.

5 Ausblick und weiteres Vorgehen

Als folgerichtiger Schritt ist nun der Funktionsbeweis anhand einer Implementierung des Systems auf einem geeigneten Demonstrator durchzuführen. Hierfür ist ein detaillierter Systementwurf zu erstellen. Ferner sind offene Fragen bezüglich der Wissensorganisation und des Wissensmanagements zu beantworten. Insbesondere ist der Regelsatz zwecks Inferenz zu erstellen.

Literatur

1. Khatib, O.: A unified approach for motion and force control of robot manipulators: The operational space formulation. IEEE Journal of Robotics and Automation **3**(1) (1987) 43–53
2. Sentis, L., Khatib, O.: Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. International Journal of Humanoid Robotics **2**(4) (2005) 505–518
3. Khatib, O., Sentis, L., Park, J.H.: A unified framework for whole-body humanoid robot control with multiple constraints and contacts. In: European Robotics Symposium 2008. Band 44 der Reihe Springer Tracts in Advanced Robotics. Springer (2008) 303–312
4. Lenz, C., Rickert, M., Panin, G., Knoll, A.: Constraint task-based control in industrial settings. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. (2009) im Druck.
5. Gibson, J.J.: The theory of affordances. In Shaw, R., Bransford, J., Hrsg.: Perceiving, Acting, and Knowing. Erlbaum, Hillsdale, NJ (1977)
6. Norman, D.: The Design of Everyday Things. Perseus Books Group (2002)
7. de Mello, L.H., Sanderson, A.C.: And/or graph representation of assembly plans. IEEE Transactions on Robotics and Automation **6**(2) (1990) 188–199
8. Sanderson, A.C., de Mello, L.H., Zhang, H.: Assembly sequence planning. AI Magazine **11**(1) (1990) 62–81
9. Featherstone, R.: Rigid Body Dynamics Algorithms. Springer, New York, NY (2008)