Repräsentation und Generalisierung von diskreten Ereignisabläufen

YORCK VON COLLANI

1 Einleitung

Das Teilprojekt D5 hat u.a. das Ziel den Situierten Künstlichen Kommunikator (SKK) mit der Fähigkeit auszustatten Montagesequenzen zu erlernen und zu generalisieren. Auf der Basis der Aufzeichnung und Analyse der konkreten Durchführung von Montageoperationen soll der SKK über immer längere Strecken und für eine immer größere Zahl von Aggregaten autonom handeln. Es soll ein prozedurales Wissen aufgebaut werden auf dessen Inhalt zurückgegriffen wird, wenn schon bekannte Baugruppen montiert werden sollen. Im Fokus stehen aber nicht der Aufbau eines prozedrual motorischen Gedächtnisses oder der Erwerb von sensomotorischen (perzeptuell-motorischen) Fertigkeiten, sonderen es werden Bewegungsaufläufe auf abstrakterer Ebene betrachtet. Die Fragestellung zur Erlernung von sensomotorischen Bewegungsprimitiven (perzeptuell- motorische Fertigkeiten) wird nicht betrachtet und das Vorhandensein von entsprechenden Fertigkeiten ausgegangen[1] Der SKK ist damit in der Lage, die Montage von unbekannten Baugruppen zu erlernen und später autonom durchzuführen.

Grundlage für die eigenständige Planung und nachfolgende Ausführung von längeren Ablauffolgen zur Konstruktion von Aggregaten sind prototypische Durchführungen von Montagesequenzen und die Benennung der Ergebnisse durch den Instrukteur. Die spätere autonome Ausführung einer Montagehandlung erfordert während dieser prototypischen Durchführungen die Aufzeichnung der Aktionsfolgen und der korrespondierenden Umwelt- und Objekterkennung in einer geeigneten Repräsentationsform. Diese Aufzeichnung kann als Akquisition von episodischen Wissen angesehen werden (wann wurde was unter welchen Bedingungen ausgeführt).

Aus diesem episodischen Wissen wird in einem nachfolgenden Verarbeitungsschritt Wissen aufgebaut, welches den SKK in die Lage versetzen soll zu einem späteren Zeitpunkt dieselbe Montagetätigkeit autonom durchführen zu können. Mit Hilfe der unterschiedlichen prototypischen Montagehandlungen werden Montagesequenzen generiert, wobei versucht wird, eine Korrelation zwischen möglicherweise unterschiedlichen Handlungsabfolgen und den dazu korrespondierenden Sensordaten zu ermitteln. Somit ist das episodische Wissen notwendig, um später prozedurales Wissen ableiten und situiert handeln zu können.

In wie weit die Generalisierung des episodischen Wissens der Aufbau eines prozeduralen Gedächtnisses bezeichnet werden kann, ist Interpretationsache. So weißt das hier beschriebene technische System einige der in [Eys94, uHH95] aufgeführten Merkmale eine prozeduralen Gedächtnisses auf:

[1] siehe auch Abschnitt 3.

- Die Fertigkeiten sind das Ergebnis von umfangreichem Training.
- Das Leistungszuwachs nimmt mit zunehmender Übung ab.
- Die erworbenen Fertigkeiten werden genau und korrekt ausgeführt (außer es kommt zu Fehlerfällen).
- Sie werden rasch und nur unter Einsatz der wirklich benötigten Ressourcen ausgeführt.

Als Inhalt des prozeduralen Gedächtnisses werden aber perzeptuell- motorische Fertigkeiten (oder allgemeiner kognitive Fertigkeiten) angesehen, die nach dem Erwerb vom Menschen *automatisiert* oder *unbewusst* verwendet werden. Da aber bei einem derzeitigen technischen System im Grund alles automatisiert ist und auch nicht von einem Bewusstsein gesprochen werden kann, ist eine Analogie vorerst nicht ersichtlich. Wird die in [uHH95] ausgeführte Erläuterung herangezogen, dass mit den Begriffen *unbewusst* und *automatisiert* vor allen ausgedrückt werden soll, dass einmal erworbene Fähigkeiten ohne bemerkenswerte kognitive Anstrengung erbracht werden, so kann von einem Aufbau eines prozeduralen Gedächtnisses gesprochen werden, da das Montagesystem später deutlich weniger Ressourcen benötigt als beim Erwerb der Fertigkeiten.

Im Folgenden wird der in [vC01] realisierte Ansatz eines lernenden Montagesystems für eine Zweiarm-Roboterkonfiguration vorgestellt. Die hier vorgestellte technische Realisierung befasst sich mit einer höheren Abstraktionsstufe als in den sonst üblichen Arbeiten über *Programming by Demonstration*. Wie schon oben erwähnt, wird davon ausgegangen, dass elementare Montageoperationen, wie in [Kai96] akquiriet werden, zur Verfügung stehen und es wird die Montage einer komplexen Baugruppe erlernt werden.

Es wird kurz auf die Repräsentation des episodischen Montagewissens eingegangen. Anschließend werden die prototypische Durchführung der Montagehandlung und die Ableitung und Generalisierung des prozeduralen Wissen erläutert.

2 Repräsentation

Die Repräsentation basiert auf dem im Teilprojekt D4 und D5 eingesetzen Montagesteuerungssystem *OPERA* [ZvCK99]. Es wird sich vieler Mechanismen, die durch dieses System zur Verfügung gestellt werden, bedient, weshalb kurz auf die beiden wichtigsten Bestandteile des Systems eingegangen wird.

2.1 Module

Über Module wird das Montagesteuersystem in sehr verschiedener Weise erweitert, wobei eine objektorientierte Sichtweise verwendet wird. Jedes Modul repräsentiert eine Aktion oder Operation innerhalb des Montagesystems. Dies kann z.B. eine einfache

Bewegungs des Manipulators oder aber auch komplexe sensomotorsiche (perzeptuell-motorische) Operationen wie z.B. das Schrauben sein. Die Kombination aus einem Modul und einem Parametersatz für die im Modul definierte Operation wird als Kommando bezeichnet.

2.2 Sequenzen / Skriptinterpreter

Eine Sequenz von solchen Kommandos bildet dann ein Skript. Skripte sind ebenfalls parametrisierbar und es besteht die Möglichkeit in ihnen, zusätzlich zu Kommandoaufrufen, einfache Verzweigungen und Aufrufe weiterer Skripte zu generieren. Ein Skript wird durch das Erstellen einer Sequenz von Kommandos gebildet, welche vom Skriptinterpreter (SI) des Montagesystems interpretiert wird.

Über solche Skripte lassen sich allgemein Montagesequenzen repräsentieren und weiterverarbeiten. Sie dienen als Basis für die Speicherung von Montagefolgen und für den Aufbau des episodischen und prozeduralen Montagegedächtnisses.

3 Diskrete Ereignisabläufe

Im Gegensatz zu dem in [Mos00] vorgestellen Ansatz, bei dem die Montagesequenzen durch die Dekomposition der zu montierenden Baugruppe aufgestellt werden, beschreibt im SFB 360 ein Instrukteur über sprachliche Anweisungen den Montagevorgang, welcher durch das Montagesystem online durchgeführt wird. Die Anweisungen werden entsprechend ihrer zeitlichen Reihenfolge gespeichert. Ist die Instruktion der Montage abgeschlossen, so werden die Montageanweisungen dauerhaft abgelegt und können zu einem späteren Zeitpunkt wiederholt werden. Für ein einfaches Aggregat, z.B. ein Leitwerk ohne Ausrichten der Leiste, wird ein Montagedurchgang genügen. Für komplexere Aggregate werden dem System mehrere und unterschiedliche prototypische Beispiele instruiert werden müssen. Hat das Montagesystem genügend Beispiele für ein spezielles Aggregat gesammelt, kann aus diesen Beispielen und den in ihnen enthaltenen Daten weiteres Montagewissen abgeleitet werden.

Die Anweisungen des Instrukteurs an das System umfassen nicht nur direkte Bewegungsanweisungen[2] an die Manipulatoren, sondern hauptsächlich Instruktionen wie: *Greife eine gelbe Schraube* oder *Schraube die Schraube in den Würfel* . Dies setzt voraus, dass das Montagesystem a priori Montagekenntnisse besitzt, um Äußerungen wie *Schrauben* und *Greifen* mit einer begrenzten Autonomie umsetzen zu können. Das realisierte System kann im ungelernten Zustand folgende Anweisungen durchführen:

Greifen: Das Greifen eines Objektes.

Stecken: Das Stecken einer Leiste oder eines Schraubwürfels auf eine Schraube.

[2] Z.B.: “Fahre Manipulator 1 an die Position xyz”.

Schrauben: Das Schrauben einer Schraube in einen Schraubwürfel oder in eine Rautenmutter.

Ablegen: Das Ablegen eines Objektes oder Aggregates auf dem Montagetisch.

Zu diesen komplexen Operationen kann das Montagesystem folgende direkte Bewegungsanweisungen (motorische bzw. sensormotorische Operationen) verarbeiten; z.B: *Einstellen einer definierten Kraft am Endeffektor, Wechsel der Endeffektororientierung, Öffnen und Schließen der Greifer, Positionierung über einem Bauteil u.s.w.*

Das Montagesystem besitzt also eine Auswahl von direkten und komplexen Anweisungen, mit deren Hilfe der Instrukteur das Montagesystem anleitet.

3.1 Generierung

Zur Repräsentation der Montagefolge bedient sich das System der in Kapitel 2.2 vorgestellten Sequenzen bzw. Skripten. Darüber hinaus besitzt das Montagesystem eine interne Repräsentation der Montageszene. Wird dem System eine Anweisung gegeben, so werden nacheinander folgende Schritte durchgeführt:

1. Prüfen, ob die Anweisung durchgeführt werden kann. Das System versucht im Rahmen seiner Möglichkeiten zu prüfen, ob eine Aktion durchführbar bzw. mit dem aktuellen Kontext stimmig ist. Ist die Anweisung eine direkte Bewegungsanweisung, so ist das System nicht in der Lage zu beurteilen, ob dies eine sinnvolle Aktion ist und führt sie ohne Prüfung aus.

2. Ist die Operation **nicht** fehlgeschlagen, so wird sie als Anweisung inkl. des vorher gespeicherten Roboterzustandes in einer Sequenz abgelegt.

3. Als nächster Schritt wird ein spezielles Modul aufgerufen, das aufgrund der durchgeführten Operation die interne Szenenrepräsentation aktualisiert. War die Operation eine direkte Bewegungsanweisung an den Roboter, so ist dies nicht immer exakt möglich. Auch dieser Aufruf wird als eine Sequenzanweisung gespeichert.

Das Ergebnis ist eine durch den Instrukteur instruierte und als Sequenz abgespeicherte und auch durchgeführte Montagesequenz, die in Form eines *OPERA*-Skriptes gespeichert wird. Eine einzelne Anweisung des Instrukteurs wird dabei als Instruktionen einer Sequenz abgelegt. eine solche Anweisung führt eine Montageoperation sowohl real als auch virtuell[3] durch. Das Montagesystem ist somit in der Lage, eine gelernte Montagesequenz simulativ durchzuführen, indem es die reale Durchführung der Anweisung auslässt.

[3] Nachführen der interen Repräsentation.

3.1.1 Interne Repräsentation des Roboterzustandes

Würde nur die Abfolge der verschiedenen Roboteroperationen als eine Sequenz abgespeichert werden, so ist die Möglichkeiten des Systems, diese Sequenz weiter zu verarbeiten, sehr eingeschränkt. Weitergehende Operationen als nur das einfache Wiederholen der Handlung sind nicht möglich, da das Montagesystem keinerlei Informationen besitzt, was seine Handlungen bewirken. Der direkte Ansatz um eine Handlung und deren Wirkung herauszufinden, ist die Beobachtung der Handlung und der Veränderungen, die diese Handlung in der Szene bewirkt. Dies setzt aber eine leistungsstarke Bildverarbeitung voraus, die nicht nur Bauteile und Aggregate auf dem Montagetisch, sondern auch in den Greifern der Manipulatoren erkennt. Ein solches Bildverarbeitungssystem stand nicht zur Verfügung und wurde in [vC01] durch die Simulation der Handlung ersetzt. Ausgehend von dem Wissen über die Semantik der verfügbaren Anweisungen, wird die reale Handlung virtuell nachvollzogen, so dass das Montagesystem zu jedem Zeitpunkt eine interne Repräsentation der Szene besitzt. Diese beinhaltet nicht nur die auf dem Montagetisch liegenden Bauteile, sondern auch den Greiferinhalt.

Solange sich die Simulation der Handlungsanweisungen auf *Greifen, Schrauben, Stecken, Ablegen*, deren Semantik bekannt ist, beschränkt, ist es möglich, die interne Repräsentation mit der Realität weitestgehend identisch zu halten. Schwierigkeiten treten dann auf, wenn direkte Bewegungsanweisungen wie z.B Relativbewegungen an die Manipulatoren gegeben werden. Deren Auswirkungen sind nicht leicht zu rekonstruieren, wenn Bauteile durch die Operation bewegt sein sollten.

Die interne Repräsentation beruht auf einem Objektmodell der *Baufix*-Bauteile das während der Arbeit zu [Fer02] entstanden ist. Es ist ein erweitertes relationales Baugruppenmodell, das die Topologie der Baugruppe repräsentiert. Es modelliert zur Zeit folgende Bauteile: Schrauben, Schraubwürfel, Leisten, Rautenmuttern und die entsprechenden Verbindungen zwischen diesen Bauteilen.

Mit Hilfe dieser Modellierung kann nicht nur eine interne Repräsentation der gegriffenen Bauteile und Aggregate erstellt, sondern können auch einfache Plausibilitätsabfragen [4] über die Durchführbarkeit von Operationen getätigt werden. Durch die Modellierung ist das Montagesystem später in der Lage zu entscheiden, ob bestimmte Aggregate oder Teile von diesen schon einmal gebaut worden sind und ob es sich schon innerhalb eines bekannten Montageplans befindet. In der Realisierung der vorliegenden Arbeit werden sowohl die Bauteile und Aggregate auf dem Montagetisch als auch die in den Greifern modelliert.

3.1.2 Sensortrajektorie

Im Regelfall wird der Instrukteur mehrere Beispiele für einen Montagevorgang geben, die vom System aufgezeichnet werden. Da für den Bau ein und desselben Aggrega-

[4] Es wird überprüft, ob eine bestimmte Montageooeration prinzipiell Ausgeführt werden: z.B: kann eine weitere Leiste auf eine Schraube aufgesteckt werden. Rahmenbedingugen, wie z.B. die Kinematik der Roboter, werden dabei nicht betrachtet.

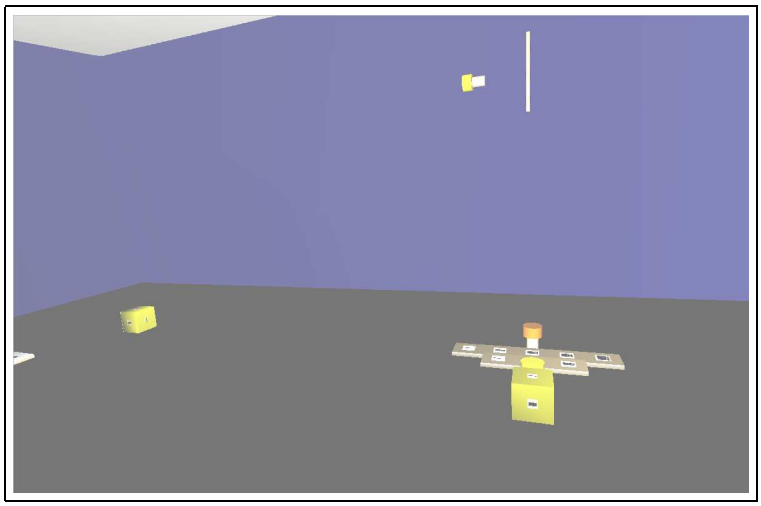

Abbildung 1: *Beispielvisualisierung eines internen Zustandes. Die Abbildung zeigt mehrere auf dem Montagetisch liegende Bauteile und eine Leiste und eine Schraube, die von den (nicht dargestellten) Robotern gehalten werden.*

tes womöglich unterschiedliche Handlungsstränge gezeigt werden, muss das Montagesystem Informationen erhalten, woran es die unterschiedlichen Montagefolgen später unterscheidet. Die einzigen Informationen, die das System zusätzlich akquirieren kann, sind Sensordaten.

Während des Lernvorgangs speichert das Montagesystem alle ihm zur Verfügung stehenden Sensordaten zusätzlich zu den Anweisungen des Instrukteurs auf. Somit besteht die gespeicherte Sequenz nicht nur aus Anweisungen an den Roboter, sondern zu jeder Anweisungen sind zusätzlich die Sensordaten, die zum Zeitpunkt **vor** der Ausführung der jeweiligen Operation gültig waren, abgelegt worden.

4 Generalisierung

4.1 Arten der Generalisierung

Es werden zwei Arten der Generalisierung behandelt.

1. Zusammenführen von gelernten unterschiedlichen Montagesequenzen für ein Aggregat bzw. eine Baugruppe.

2. Anwendung von bestehenden gelernten Montagesequenzen mit Variierung der verwendeten Bauteile.

Im Regelfall werden mehrere Beispiele für den Zusammenbau eines Aggregates gegeben. Dabei ist nicht zu erwarten, dass die resultierenden Montagesequenzen identisch ausfallen. Zwei Ursachen kommen in Betracht; die Unterschiede ergeben sich durch

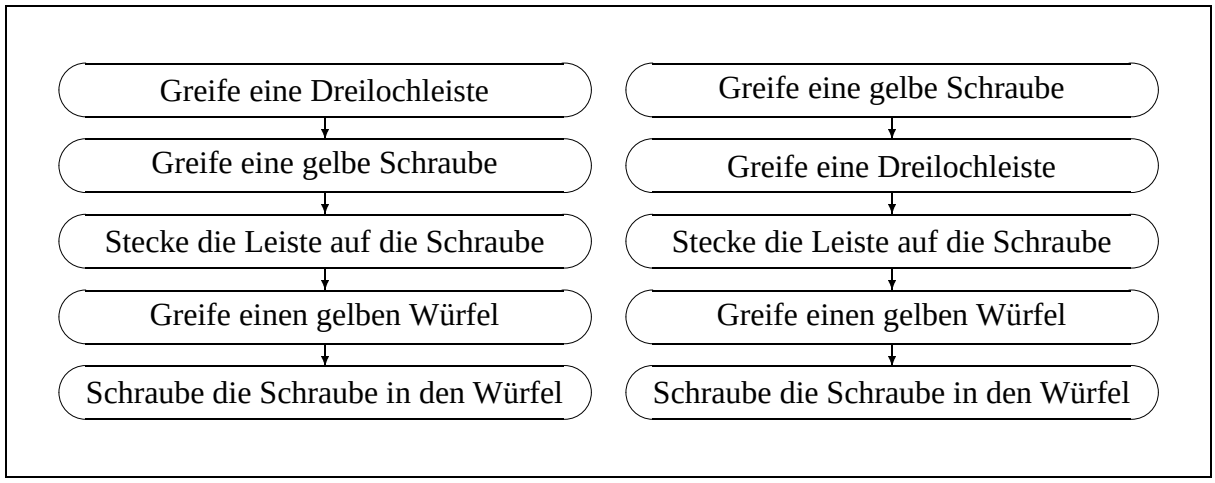

Abbildung 2: *Flussdiagramm zweier Sequenzen zum Bau eines Leitwerks.*

unterschiedliche Szenarien und Vorkommnisse während der Montage oder durch das Vorhandensein verschiedener wählbarer Wege zum Montageziel. Abb. 2 zeigt zwei unterschiedliche Sequenzen, die beide den Bau desselben Aggregates beschreiben.

Da a priori nicht bekannt ist, welche der vorhandenen Montagesequenzen bei einer späteren Wiederholung gewählt werden müssen, sind die vorhanden Sequenzen zu fusionieren und so zu ändern, dass das Montagesystem selbstständig feststellen kann, wann welche Variante der Sequenz ausgeführt werden muss.

Ist die Montage eines Aggregates vom System erlernt worden, so sollen diese Sequenzen auch mit leicht unterschiedlichen Bauteilen zurechtkommen. Ist z.B. der Bau eines Leitwerks mit einer gelben Schraube, einer Dreilochleiste und einem Schraubwürfel erlernt worden, so ist es nur ein kleiner Unterschied, ob dies nun mit einer blauen Schraube gebaut wird. Es muss eine blaue statt einer gelben Schraube gegriffen werden.

4.2 Zusammenfassen von Sequenzen

4.2.1 Vergleichsmethode

Zwei Kommandos $u, v \in C_M$ sind gleich, wenn sowohl die Zustandstransformationsfunktionen M_u, M_v als auch die Parameter p_u, p_v identisch sind. Analog hierzu sind zwei Sequenzaufrufe identisch, wenn ihre Parameter identisch sind. Zwei *IF*-Instruktionen (Operationen aus der Menge C_2 des SI) werden **immer** als unterschiedlich betrachtet. Zwei Sequenzen π und ρ sind identisch, wenn

$$\pi(z) = \rho(z) \quad \forall\, z \quad \Longleftrightarrow \quad \pi = \rho$$

Allgemeine Vorgehensweise Sind mehr als zwei Sequenzen zusammenzufassen, wird als erster Schritt die zweite Sequenz mit der ersten fusioniert. Hierfür werden die Instruktionen Position für Position miteinander verglichen. Als nächster Schritt wird die dritte Sequenz mit der nun modifizierten ersten zusammengefasst u.s.w. . Die hinzuzufügende Sequenz wird nachfolgend Quellsequenz ρ und die Sequenz, in die alle Sequenzen eingefügt werden, Zielsequenz π genannt. Sind keine Unterschiede zwischen Ziel-und Quellsequenzen vorhanden, so werden nur die Sensordaten der jeweiligen Instruktionen in die Zielsequenz übertragen. Eine Zielsequenzinstruktion besitzt anschließend nicht nur ein Beispiel für den Sensorzustand des Systems, sondern mehrere.

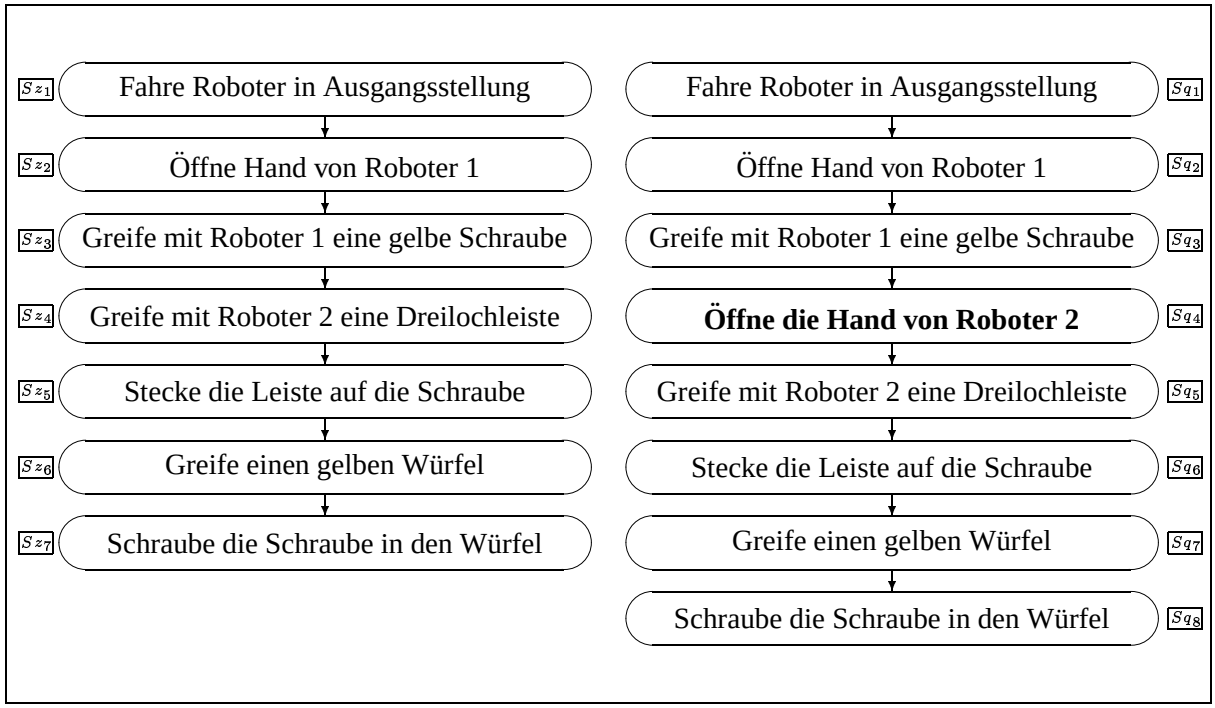

Abbildung 3: *Beispiel für zusätzliche Instruktionen.* Sz_i *und* Sq_i *repräsentieren die aufgezeichneten Sensorzustände der* i*-ten Instruktion.*

4.2.2 Erzeugen von Verzweigungen

Unterscheiden sich zwei oder mehrere Sequenzen obwohl das Endprodukt ein- und dasselbe Aggregat ist, so gibt es zwei Gründe hierfür:

1. Die Reihenfolge der Operationen ist teilweise beliebig und der Instrukteur hat davon Gebrauch gemacht.
2. In der realen Szene ist eine andere Situation eingetreten als bei den anderen Beispielen und daher musste die Montage mit unterschiedlichen Operationen fortgesetzt werden. Dies sind z.B. Ausrichtoperationen.

In beiden Fällen müssen Verzweigungen[5] generiert und mit dem zusätzlichen Handlungsstrang in die Sequenz eingefügt werden.

Ist eine Sequenzposition z_Q (Position in der Quellsequenz) bzw. z_Z (Position in der Zielsequenz) ermittelt worden, an der Ziel- und Quellsequenz unterschiedlich sind $\pi(z_Z) \neq \rho(z_Q)$, so wird im nächsten Schritt herausgefunden, wo die beiden zu vergleichenden Sequenzen wieder identisch sind. Im Folgenden werden nur die Teilsequenzen betrachtet, auf denen sich die Quell- und die Zielsequenzen unterscheiden. Es sind folgende Fälle zu betrachten:

1. Die Quellsequenz besitzt zusätzliche Instruktionen gegenüber der Zielsequenz (siehe Abb 3).

2. Die Zielsequenz besitzt gegenüber der Quellsequenz zusätzliche Instruktionen.

3. Beide Sequenzen unterscheiden sich in einer Teilsequenz (siehe Abb. 4).

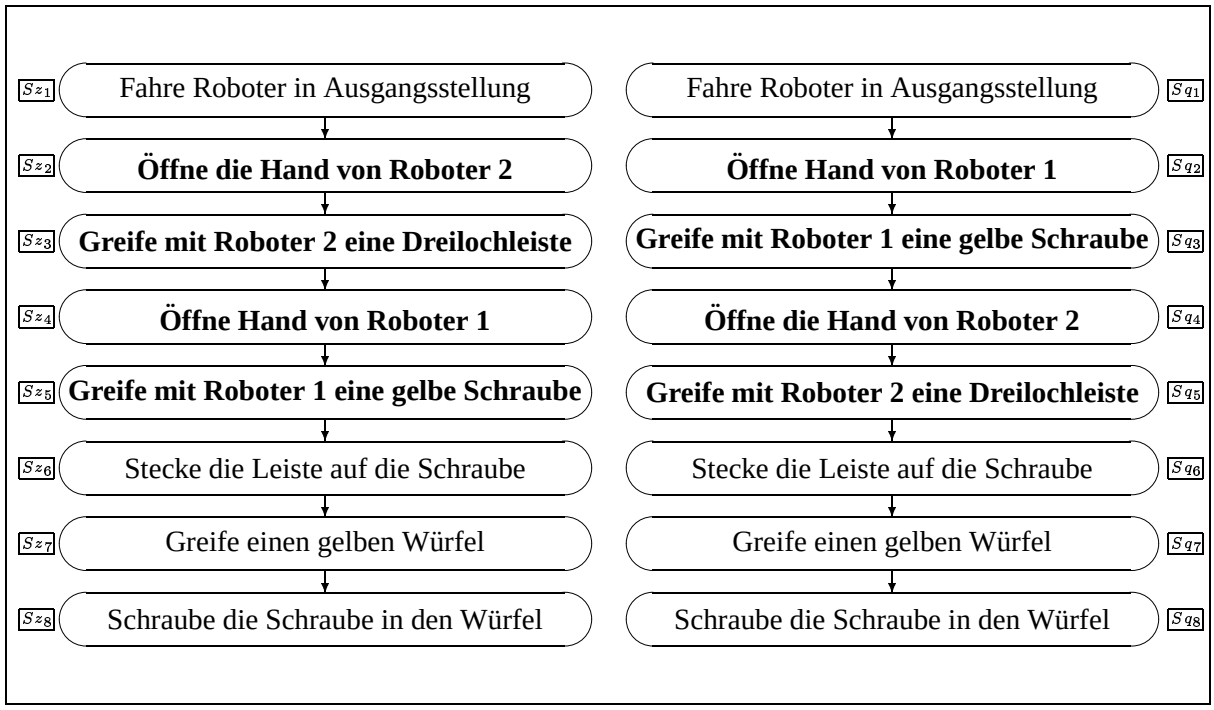

Abbildung 4: *Beispiel zweier unterschiedlicher Teilsequenzen (fett).* Sz_i *und* Sq_i *repräsentieren die aufgezeichneten Sensorzustände der* i*-ten Instruktion.*

Bevor eine Verzweigung in die Zielsequenz eingebaut werden kann, müssen erst die Positionen ermittelt werden, ab der die Quell- und Zielsequenz wieder identisch sind.

[5]Operationen aus der Menge C_2 des SI.

Daher wird erst geprüft, ob folgende Aussage gilt [6]:

$$\exists\, i \geq z_Z \ \wedge \ j \geq z_Q \quad \text{für das gilt} \quad \pi(i+m) = \rho(j+m) \quad \text{mit} \quad m = 0, \ldots, b \qquad (1)$$

Erst wenn mehrere aufeinanderfolgende Instruktionen identisch sind ($b >= 4$), wird davon ausgegangen, dass die nachfolgenden Instruktionen zu einer größeren identischen Teilsequenz gehören. Bei einer kleineren Anzahz von übereinstimmenden Instruktionen, ist eine Identität nur zufällig.

η bezeichnet die neu entstehende Sequenz, welche vor dem Einfügen der Instruktionen aus ρ mit π identisch ist und nach dem Einfügen zur neuen Zielsequenz $\eta \longrightarrow \pi$ wird. Nach dem Finden der Position von der an die Ziel- und Quellsequenz wieder identisch sind, kann die Verzweigung in die Zielsequenz eingebaut werden. Hierzu wird zuerst an der Position z_Z der Zielsequenz eine *IF*-Instruktion eingefügt, deren boolsche Funktion im Bedingungsteil vorerst undefiniert bleibt. Damit ist $\eta(z_Z + 1) = \pi(z_Z)$.

Fall 1 Besitzt die Quellsequenz ρ gegenüber der Zielsequenz π zusätzlich Instruktionen, so müssen diese in die Zielsequenz ab der Position z_Z eingebaut werden. Nach dem Einbau ist die neue Zielsequenz η wie folgt definiert:

$$\begin{array}{lcl}
\eta(k) & = & \pi(k) \quad \text{für} \quad k = z_Z - 4, \ldots, z_Z - 1 \\
\eta(z_Z) & = & \tau_{\mathsf{IF}} \\
\eta(z_Z + 1) & = & \rho(z_Q) \\
\eta(z_Z + 2) & = & \rho(z_Q + 1) \\
 & \vdots & \\
\eta(z_Z + (j - z_Q) + 1) & = & \rho(j - 1) \\
\eta(z_Z + (j - z_Q) + 2) & = & \pi(z_Z) \\
\eta(z_Z + (j - z_Q) + 3) & = & \pi(z_Z + 1) \\
 & \vdots &
\end{array}$$

τ_{IF} ist die eingefügte *IF*-Instruktion $\tau_{\mathsf{IF}} \in C_2$.

Fall 2 Für den Fall, dass die Quellsequenz weniger Instruktionen als die Zielsequenz besitzt, muss nur eine *IF*-Instruktion an Position z_Z eingefügt werden.

Fall 3 Für den Fall, dass Ziel- und Quellsequenz unterschiedliche Handlungsstränge und damit unterschiedliche Teilsequenzen besitzen, muss ein *IF-THEN-ELSE* Konstrukt eingebaut werden. Als erster Schritt wird an Position z_Z die *IF*-Instruktion eingefügt; anschließend die Instruktionen $\rho(z_Q), \ldots, \rho(j)$ und eine Sprungoperation. Die resultierende

[6] Es ist zu beachten, dass hier mehrere mögliche algorithmische Umsetzungen möglich sind, die u. U. unterschiedliche Resultate liefern.

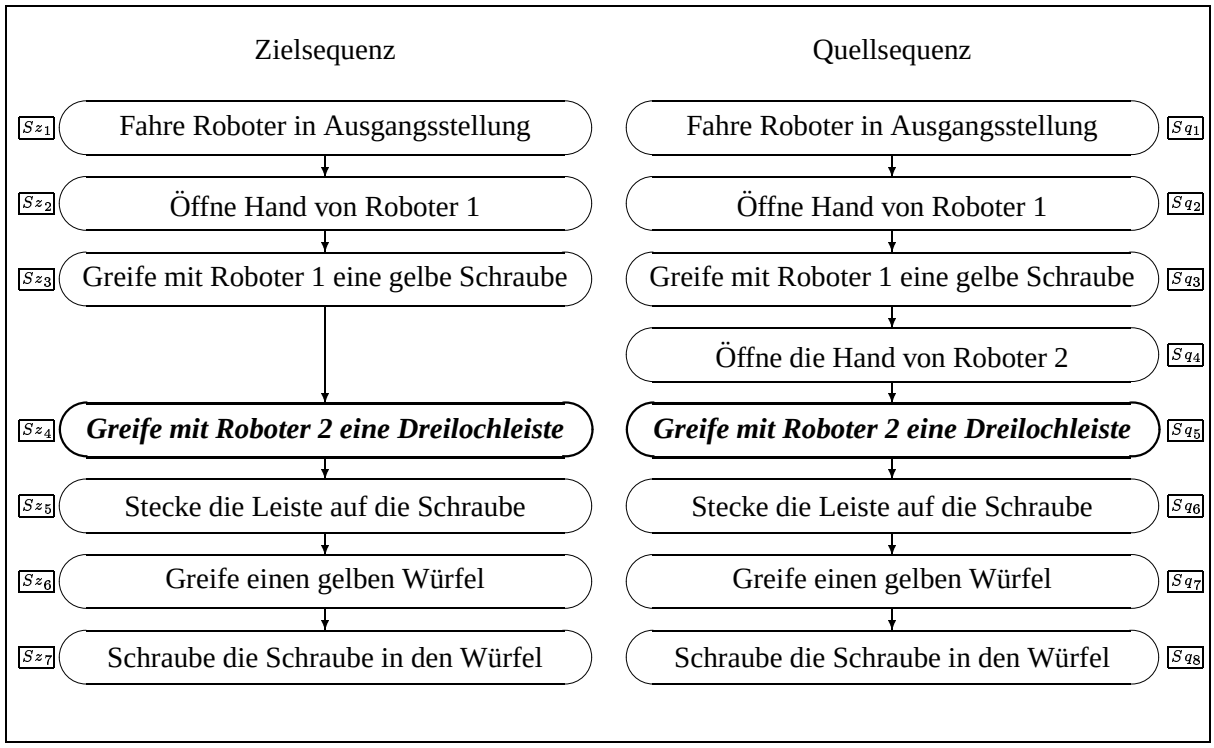

Abbildung 5: *Erfüllung der Bedingung für Fall 1.* Sz_i *und* Sq_i *repräsentieren die aufgezeichneten Sensorzustände der* i*-ten Instruktion.*

Teilsequenz der neuen Sequenz η setzt sich dann wie folgt zusammen:

$$
\begin{array}{lcl}
\eta(k) & = & \pi(k) \quad \text{für} \quad k = z_Z - 4, \ldots, z_Z - 1 \\
\eta(z_Z) & = & \tau_{\mathsf{IF}} \\
\eta(z_Z + 1) & = & \rho(z_Q) \\
 & \vdots & \\
\eta(z_Z + j - z_Q) & = & \rho(j - 1) \\
\eta(z_Z + j - z_Q + 1) & = & \tau_{\mathsf{Jump}} \\
\eta(z_Z + j - z_Q + 2) & = & \pi(z_Z) \\
 & \vdots &
\end{array}
$$

mit $\tau_{\mathsf{IF}} \in C_2$, $\tau_{\mathsf{Jump}} \in C_1$.

Sind die Verzweigungen eingebaut, so müssen abschließend die Sensordaten von $\rho(z_Q)$ als auch von $\pi(z_Z)$ der bei z_Z eingefügten *IF*-Instruktion hinzugefügt werden. Sowohl die Sensordaten von $\pi(z_Z) = \eta(z_Z + (j - z_Q) + 2)$ als auch die von $\rho(z_Q)$ beschreiben vor der Verzweigung mögliche Sensorzustände. Abb. 7 zeigt die resultierende Sequenz für das Beispiel aus Abb.6.

Berücksichtigung von bestehenden *IF*-Instruktionen Ein Sonderfall ist gegeben, wenn Teilsequenzen der Ziel- und die Quellsequenz nur deshalb nicht identisch sind, weil in der Zielsequenz schon eine Verzweigung eingebaut worden ist. Da z.Z. der Instrukteur

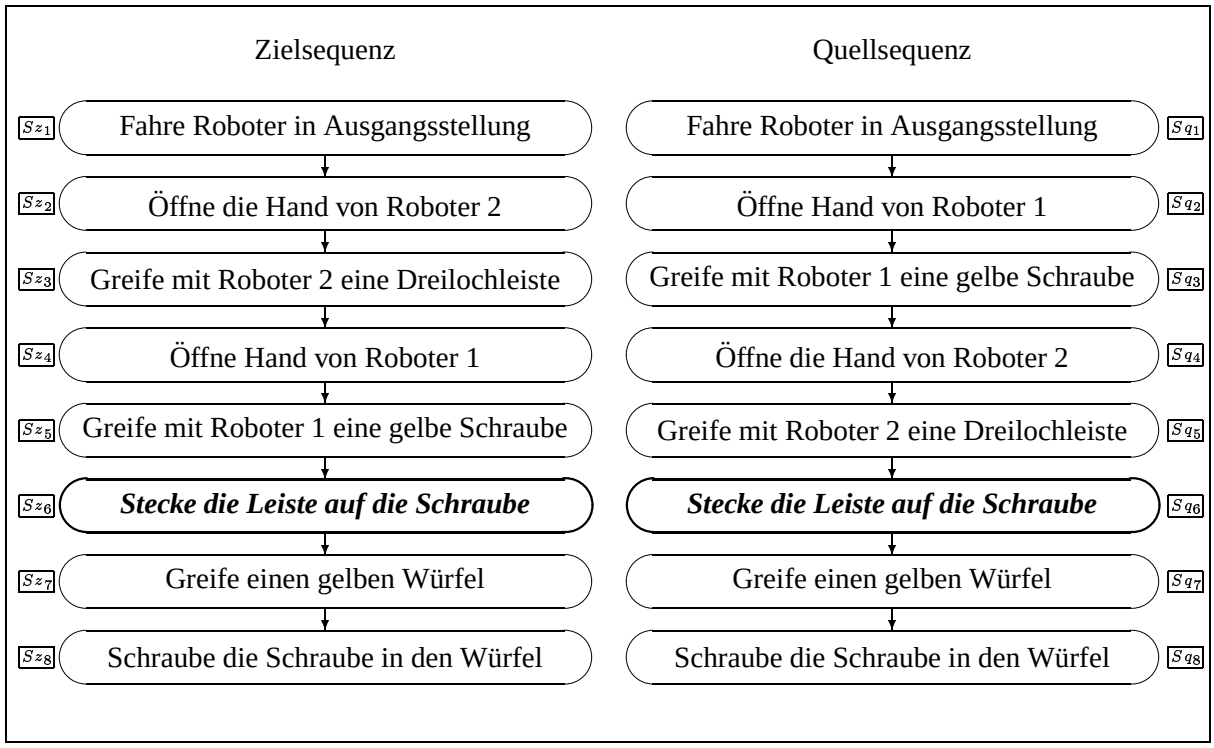

Abbildung 6: *Erfüllung der Bedingung für Fall 3.* Sz_i *und* Sq_i *repräsentieren die aufgezeichneten Sensorzustände der* i*-ten Instruktion.*

keine Anweisungen geben kann, die Bedingungen enthalten, können solche Instruktionen in der Quellsequenz nicht auftreten. Tritt die Situation auf, dass $\pi(z_Z) \neq \rho(z_Q)$ und $\pi(z_Z) \in C_2$, dann muss überprüft werden, ob die Teilsequenz

$$\rho(i) \quad \text{für} \quad i = z_Q, \ldots, j$$

identisch mit der Teilsequenz des *THEN*-Abschnittes oder, soweit existent, identisch mit dem *ELSE*-Abschnitt der Zielsequenz ist. Ist dies der Fall, so werden nur die beim Lernprozess gespeicherten Sensordaten in der Quellsequenz in die entsprechenden Instruktionen der Zielsequenz übernommen und die Zielsequenz wird ansonsten unverändert gelassen. Ist die Teilsequenz nicht mit dem *IF*- oder *THEN*-Abschnitt der Zielsequenz identisch, so wird wie oben beschrieben eine Verzweigung aufgebaut.

4.2.3 Auswertung von Sensormustern

Sind alle verfügbaren Sequenzen für die Montage eines Aggregates zusammengefasst, müssen die eingefügten *IF*-Instruktionen vervollständigt werden. Die boolsche Funktion dieser Instruktionen ist noch undefiniert. Die jeweilige Entscheidung, ob bei einer Verzweigung der eine oder der andere Handlungsstrang genommen wird, kann das

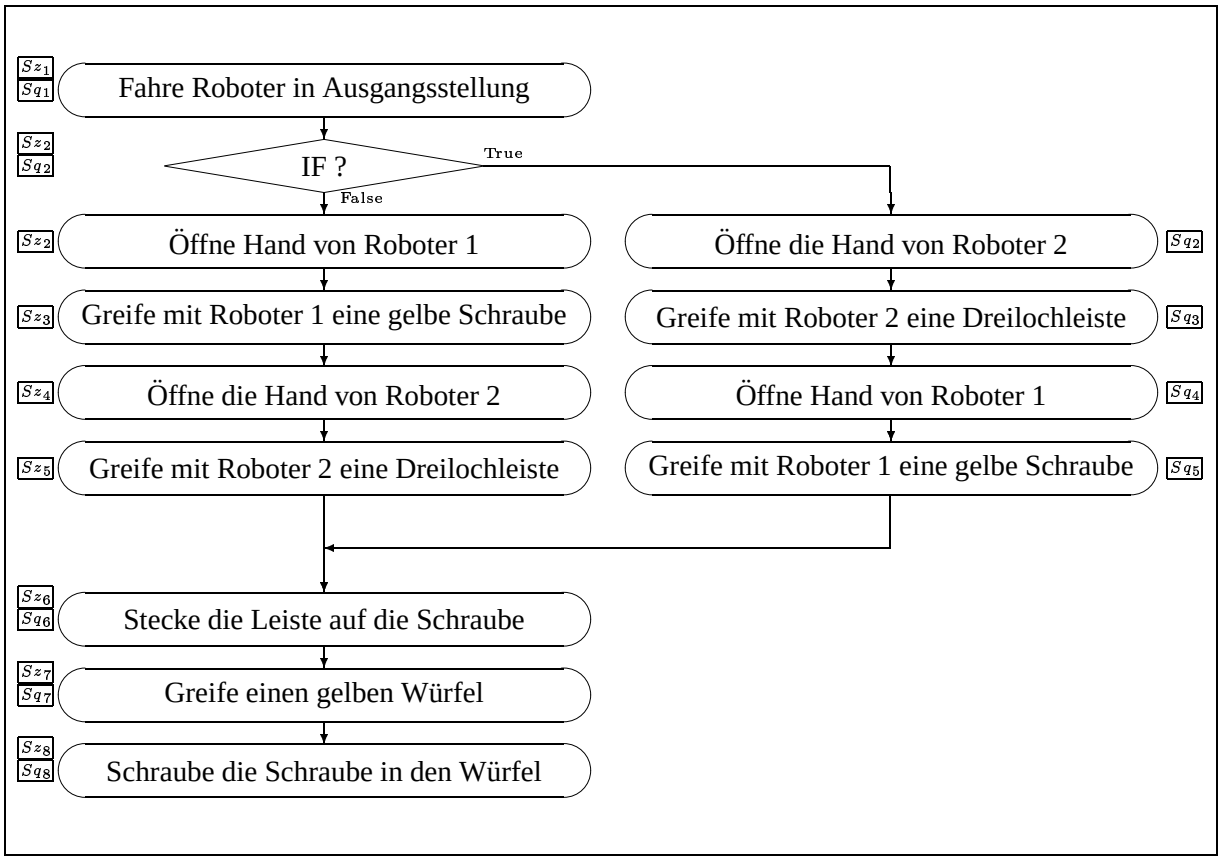

Abbildung 7: *Flussdiagramm der resultierenden Sequenz für Fall 3.* Sz_i *und* Sq_i *repräsentieren die abgelegten Sensorzustände der* i*-ten Instruktion. Es ist zu erkennen, wie die aufgezeichneten Sensordaten nach der Zusammenlegung von zwei Sequenzen zugeordnet werden. Der* IF*-Instruktion werden die Sensordaten der möglichen Nachfolgeinstruktion zugeordnet, da beide Zustände vor dieser Verzweigung aufgetreten sind.*

Montagesystem nur anhand der aufgezeichneten Sensordaten (siehe Abschnitt 3) ausfindig machen. Es wird daher ermittelt, welche der zur Verfügung stehenden Sensoren für eine Entscheidung zu berücksichtigen sind.

Beim Aufbau eines flexiblen Montagesystems hat sich die Verwendung eines B-Spline Fuzzy Reglers [ZK96, ZL96] in verschiedenen Anwendungen bewährt. In [ZFK00, ZF98, ZvCK97] wird dieser Ansatz als Regler zur Steuerung von beschriebenen Manipulatoren verwendet. Schraub-, Steck- und Mehrarmtrageoperationen sind damit verwirklicht worden. Bei diesen Operationen ist die Fähigkeit des B-Spline Fuzzy Reglers, sich schnell und insbesondere während des Betriebes zu adaptieren, ein großer Vorteil. Die Verwendung von Reglerparametern einer Trageoperation in einem Regler für einen Schraubvorgang [FZK99] zeigt die Flexibilität des B-Spline Fuzzy Ansatzes bei gleichen physikalischen Gesetzmäßigkeiten. Es konnte gezeigt werden, dass mit Hilfe des B-Spline Fuzzy Reglers als mehrdimensionalem Funktionsapproximator sowohl

Visual Servoing [ZKS00, ZKS99, ZSK99] , z.B. das Ausrichten eines Greifer über einem Bauteil, als auch Datenfusion [vCSZK99, vCFZK00] betrieben werden kann. Diese Eigenschaft steht hier im Vordergrund.

Basierend auf dem B-Spline Fuzzy Modell wird ein Regler gebaut, der die Daten der berücksichtigten Sensoren fusioniert. Die Ausgabe des Regler ist in diesem Fall aber nicht eine Steuergröße für einen Manipulator, sondern nur ein boolscher Wert (0 oder 1) der Angibt, ob die Bedingung der IF- Instruktion erfüllt ist oder nicht.

4.2.4 Retrieval

Ist die Generalisierung der Beispielsequenzen für ein Aggregat abgeschlossen, wird die Sequenz abgelegt und kann zu einem späteren Zeitpunkt wiederholt werden. Zusätzlich zu der Handlungssequenz wird der Anfangszustand des Montagesystems mit abgelegt. Bei einem Abruf der Montagesequenz wird als erster Schritt geprüft, ob der aktuelle Anfangszustand des Systems mit dem abgelegten Zustand der Sequenz übereinstimmt. Ist dies der Fall, kann die Sequenz ausgeführt werden.

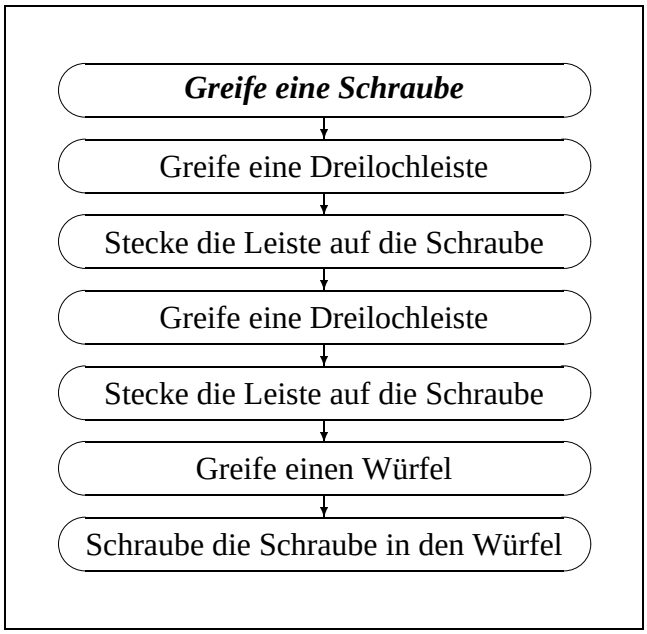

Abbildung 8: *Flussdiagramm der Beispielsequenz zum Bau eines Propellers.*

Ist dies nicht der Fall, wird auf episodisches Wissen zurückgegriffen und versucht aufgrund der in der Sequenz gespeicherten Auswirkung der Operationen auf die Szene herauszufinden, ob der aktuelle Zustand des Montagesystems einem Zwischenzustand der Montagesequenz entspricht. Die Montage wird vom System simulativ durchgegangen. Lässt sich ein solcher Zwischenzustand finden, der mit dem aktuellen Zustand übereinstimmt, wird ab dieser Stelle die Sequenz ausgeführt. Das Montagesystem ist

damit in der Lage, eine teilweise durchgeführte Montage eines ihm bekannten Aggregates autonom zu beenden.

Beispiel Soll z.B. der Propeller aus dem SFB-Szenario gebaut werden, so würde die in Abb. 8 gezeigte Sequenz ausgeführt. Es müssen zwei 3-Loch Leisten auf eine Schraube gesteckt werden, die danach in einen Schraubwürfel geschraubt wird.

Der übliche Anfangszustand des Systems wären zwei Manipulatoren mit leeren Greifern. Beginnt aber der Instrukteur den Bau eines Propellers mit Einzelanweisungen, erreicht er einen Zwischenzustand der entsprechenden Sequenz für den Bau eines Propellers. Würde nun die Anweisung gegeben, einen Propeller zu bauen, wäre die vollständige Ausführung der Sequenz zum Bau eines Propeller nicht die intendierte Reaktion. Es würde ein Abschluss der begonnenen Handlung erwartet, nicht ein Neubeginn. Die in Abb. 9 fett markierten Instruktionen müssen nicht mehr ausgeführt werden. Durch simulatives Durchgehen der Sequenz wird die entsprechende Stelle in der Sequenz gefunden, ab der mit der Ausführung der Montagesequenz begonnen wird.

Abbildung 9: *Flussdiagramm der Beispielsequenz zum Bau eines Propellers.*

4.3 Modifikation

Hat das Montagesystem den Bau eines Aggregates erlernt, so sollte es ausgehend vom Gelernten auch ähnliche Aggregate zusammenbauen können. Ein erster Schritt hierzu ist die Verwendung von unterschiedlichen Bauteilen. So soll z.B. folgende Anweisung vom Instrukteur umgesetzt werden können:

Baue den Propeller, aber mit Fünflochleisten.

Hat das System den Bau des Propellers mit zwei Dreilochleisten, erlernt, so muss es jetzt die erlernte Sequenz so **modifizieren**, dass die Anweisung entsprechend umgesetzt wird; es müssen Fünflochleisten statt Dreilochleisten gegriffen werden.

Um diese Modifikation durchzuführen, geht das Montagesystem die Sequenz in ihren unterschiedlichen Ablaufmöglichkeiten durch und sucht nach der Anweisung, die das Greifen von Leisten realisieren. Sind diese Anweisungen anhand der Instruktionen, die die interne Szenenrepräsentation modifizieren, gefunden, so wird der Greifparameter für die Lochanzahl geändert.

5 Zusammenfassung

Das Montagesystem lernt die Montage von (Teil-)Aggregaten, indem der Instrukteur dem System Schritt für Schritt über sprachliche Anweisungen die durchzuführende Handlung angibt. Die entsprechenden Instruktionen werden zusammen mit den vor der Ausführung gültigen Sensordaten als ein Skript in *OPERA* abgespeichert. Zusätzlich wird zur jeder Instruktion eine weitere Instruktion generiert, die eine interne symbolische Repräsentation der Umwelt mit der Realität abgleicht. Die durch den Lernvorgang entstandene Sequenz ist ein Beispiel für den Bau eines gewünschten Aggregates. Je mehr unterschiedliche Szenerien für den Bau eines Aggregates existieren, um so mehr prototypische Montagesequenzen werden benötigt, um in alle Situationen die Montage des Aggregates durchführen zu können.

Die zur Verfügung stehenden Beispielsequenzen werden zu einem Skript fusioniert, wodurch aufgrund unterschiedlicher Handlungsstränge in den einzelnen Beispielen Verzweigungen aufgebaut werden müssen. Anhand der in den Beispielsequenzen gespeicherten Sensordaten werden die für eine Entscheidung relevanten Daten extrahiert.

Das Endprodukt ist ein Montageskript für ein (Teil-)Aggregat, durch das das Montagesystem in der Lage ist, das Aggregat auch in unterschiedlichen Situationen autonom zusammenzubauen. Zusätzlich ist das Montagesystem fähig, durch Modifizieren des Montageskriptes das erlernte Aggregat mit anderen als den erlernten Bauteilen zu montieren.

Literatur

[Eys94] M. W. Eysenck. *Dictionary of Cognitive Psychology*. 1994.

[Fer02] Markus Ferch. *Lernen von Montagestrategien in einer Multiroboterumgebung*. PhD thesis, University of Bielefeld, 2002.

[FZK99] M. Ferch, J. Zhang, and A. Knoll. Robot skill transfer based on b-spline fuzzy controller for force-control tasks. In *In Proceedings of the IEEE International Conference on Robotics and Automation, Detroit*, ICRA, 1999.

[Kai96] Michael Kaiser. *Interaktive Akquisition elementarer Roboterfähigkeiten*. PhD thesis, Univ. Karlsruhe, 1996.

[Mos00] Heiko Mosemann. *Beiträge zur Planung, Dekomposition und Ausführung von automatischen generierten Roboteraufgaben.* PhD thesis, Tech. Univ. Braunschweig, 2000.

[uHH95] Rainer H. Kluwe und Hilde Haider. *Erwerb kognitiver Fertigkeiten durch Übung*, chapter Das Gedächtnis (Probleme - Trends - Perspektiven). Hogrefe, 1995.

[vC01] Yorck von Collani. *Repräsentation und Generalisierung von diskreten Ereignisabläufen in Abhängigkeit von Multisensormustern*. PhD thesis, Univ. Bielefeld, 2001.

[vCFZK00] Y. von Collani, M. Ferch, J. Zhang, and A. Knoll. A general learning approach to multi sensor based control using statistical indices. In *Proc. of the IEEE Int. Conf. on Robotics and Automation, San Francisco, California*, April 2000.

[vCSZK99] Y. von Collani, C. Scheering, J. Zhang, and A. Knoll. A neuro-fuzzy solution for integrated visual and force control. In *Proceedings of the International Conference on Multi sensor Fusion and Integration of Intelligent Systems, Taipeh*, Aug. 1999.

[ZF98] J. Zhang and M. Ferch. Rapid on-line learning of compliant motion for two-arm coordination. In *Proceedings of the IEEE International Conference on Robotics and Automation, Leuven, Belgium*, ICRA, 1998.

[ZFK00] J. Zhang, M. Ferch, and Alois Knoll. Carrying heavy objects by multiple manipulators with self-adapting force control. In *Proc. of Intelligent Autonomous Systems*, pages 204–211, 2000.

[ZK96] J. Zhang and A. Knoll. Constructing fuzzy controllers with b-spline models. In *Proceedings of IEEE International Conference on Fuzzy Systems*, 1996.

[ZKS99] J. Zhang, A. Knoll, and R. Schmidt. A neuro-fuzzy control model for fine-positioning of manipulators. *Journal of Robotics and Autonomous Systems, Elsevier Science*, 1999.

[ZKS00] J. Zhang, A. Knoll, and R. Schmidt. A neuro fuzzy control model for fine-positioning of manipulators. *Journal of Robotics and Autonomous System*, 32:101–113, 2000.

[ZL96] J. Zhang and K.V. Le. Self-optimization of a fuzzy controller with b-spline models. In *Proceedings of Fourth European Congress on Intelligent Techniques and Soft Computing*, 1996.

[ZSK99] J. Zhang, R. Schmidt, and A. Knoll. Appearance-based visual learning in a neuro-fuzzy model for fine-positioning of manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1999.

[ZvCK97] J. Zhang, Y. von Collani, and A. Knoll. On-line learning of b-spline fuzzy controller to acquire sensor-based assembly skills. In *Proc. of the IEEE Conf. on Robotics and Automation, Albuquerque, New Mexico*, ICRA, April 1997.

[ZvCK99] J. Zhang, Y. v. Collani, and A. Knoll. Interactive assembly by a two-arm robot agent. *Journal of Robotics and Autonomous Systems*, 29:91–100, 1999.