Skripte als ereignisorientierte Repräsentationsmechanismen in der Robotik

Jürgen Dorn, Günter Hommel, Alois Knoll

Die wissensgestützte Programmierung von Robotern ist noch nicht so weit fortgeschritten, daß sie industriell einsetzbar wäre. Die Probleme liegen dabei zum Teil in der Komplexität der realen Anwendungen. Aber auch im Bereich der künstlichen Intelligenz entwickelte Methoden (z.B. für die Planung in der Klötzchenwelt) eignen sich nicht zur Lösung komplexerer Fragestellungen. Ein Ansatz zur Überwindung der dabei auftretenden Probleme ist die Einführung strukturierter Repräsentations - mechanismen als Abstraktionshilfsmittel.

Verglichen mit anderen Bereichen der künstlichen Intelligenz existiert im Bereich der Robotik sehr viel ereignisorientiertes Wissen, d.h. es wird ein Nacheinander von Aktionen beschrieben. Prozedurale Programmiersprachen, mit denen zwar eine Folge von Aktionen beschrieben werden kann, bieten jedoch keine adäquaten Hilfsmittel zur Beschreibung von Wissen über die Umwelt von Robotersystemen.

Wir beschreiben ein Modell, in dem Skripte als Schnittstelle zwischen Planungsmodul, Überwachungs - modul, Kollisionsvermeidungsmodul und Skript-Interpretationsmodul (d.h. der Ausführung der Aktionen durch den Roboter) benutzt werden.

G. Hommel et al. (eds.), *Informatik-Anwendungen — Trends und Perspektiven*
© Springer-Verlag Berlin Heidelberg 1986

1 Deklarative Wissensrepräsentation

Die strikte Trennung von Daten, Operationen und Ablaufstruktur ist für Systeme im Bereich der künstlichen Intelligenz charakteristisch. Wissen (Daten und Operationen) wird also deklarativ beschrieben und ist damit unabhängig von seiner Verarbeitung (Ablaufstruktur) leicht an eine sich ändernde Umwelt anzupassen.

Im Bereich der Robotik und der Bildverarbeitung wurden insbesondere semantische Netze [Niem 85] und Rahmen (frames) erfolgreich eingesetzt. Mit Hilfe dieser Wissensrepräsentationsformen kann die Umwelt beschrieben werden, in der sich der Roboter bewegt.

Semantische Netze erlauben die Beschreibung beliebiger Relationen zwischen Objekten. Da geeignete Strukturierungsmechanismen für solche Netze fehlen, neigen sie dazu, bei komplexen Problemen leicht unübersichtlich zu werden. Rahmen stellen solche Strukturierungsmechanismen dar; sie erlauben die Beschreibung von Objekten und deren Eigenschaften, sodaß sie zumindest für die Repräsentation der Umwelt gut geeignet sind.

Beiden Darstellungsarten fehlt die Einbeziehung von Zeiten bzw. die Darstellung einer Folge von Ereignissen. In Rahmen können wir uns zwar Fächer (slots) vorstellen, die eine Zeit beinhalten, aber die Behandlung wird nicht durch einen vorgegebenen Mechanismus unterstützt. In der Automatisierungstechnik ist der zeitliche Ablauf von Ereignissen ein sehr wichtiger Aspekt. Es muß dargestellt werden können, ob zwei Ereignisse parallel oder nacheinander stattfinden. Für diese Darstellung benutzen wir Skripte.

Skripte stellen einen Mechanismus dar, der von Schank u.a. [Scha 77] und [Scha 81] entwickelt wurde, um kleine Zeitungstexte bezüglich der in ihnen beschriebenen *Ereignisse* zu verstehen. Dabei wurde von der deklarativen Sichtweise der Rahmen ausgegangen. Unterschiede ergeben sich aus der unterschiedlichen Interpretation der Fächer.

2 Skripte

Ein Skript stellt eine prototyphafte Beschreibung einer Folge von Ereignissen dar. Zur Spezifikation von Skripten gehören neben Ereignissen weitere Eigenschaften (properties), die erst die Leistungsfähigkeit dieses Hilfsmittels zur strukturierten Wissensrepräsentation ausmachen. Alle Eigenschaften werden in Fächern des Skriptes notiert.

Wir unterscheiden zwischen solchen Skripten, deren Ablauf irreversible Umweltzustandsänderungen zur Folge hat (Beispiel : Bohren) und solchen, bei denen die Zustandsänderung vollständig zurückgnommen werden kann (Beispiel : Bewegen). Bei den letztgenannten ist es möglich, das Skript als atomare Transaktion zu betrachten, was insbesondere für die Fehlerbehandlung die Anwendung der hierfür bekannten Strategien ermöglicht [Lamp 81]. Das bedingt allerdings, daß jedes Einzelereignis des Skriptes *rücksetzbar* ist. Sobald ein Ereignis im Skript nicht rücksetzbar ist, kann das Skript nicht mehr als Transaktion aufgefaßt werden. Die Rücksetzbarkeit eines Ereignisses muß vom Benutzer spezifiziert werden. Die grafische Darstellung eines Skriptes zeigt Bild 9.

2.1 Eintrittsbedingungen und Ergebnisse

Wichtige Eigenschaften von Skripten sind Eintrittsbedingungen und Ergebnisse. Eintrittsbedingungen beschreiben einen Zustand der Umwelt, der gegeben sein muß, damit das Skript anwendbar wird. Beschreibt ein Skript z.B. die Bewegung eines Objektes durch den Roboter, so muß die Bedingung erfüllt sein, daß das Objekt beweglich ist.

Ergebnisse beschreiben den veränderten Zustand, den die Umwelt aufweist, wenn die Ereignisse stattgefunden haben. Beschreibt das Skript den Transport eines Objektes durch den Roboter auf ein anderes Objekt, so kann das bedeuten, daß das untere Objekt nicht mehr beweglich ist.

2.2 Requisiten

Requisiten stellen ein Fach dar, in dem alle an dem Skript beteiligten Objekte aufgeführt werden. Typische Requisiten sind Werkzeuge, die ein Roboter für die Lösung einer Aufgabe benötigt. Spezielle Aufnahmen für Montagearbeiten, die man benötigt, um z.B. eine Verschraubung durchführen zu können, modellieren wir ebenso als Requisit. Requisiten werden auch zur Fehlerbehandlung eingesetzt. Rutscht ein veröltes Objekt aus dem Greifer, kann das Requisit Öl Stichwort für ein Skript sein, das das Öl vom Objekt entfernt.

2.3 Rollen

Rollen sind Fächer für handelnde Objekte, die eine Zustandsänderung bewirken können, wie z.B. Roboter, Fließbänder und andere Maschinen. Wir sehen die wissensgestützte Programmierung in einem größeren Zusammenhang und betrachten deshalb neben dem Roboter weitere Rollen wie z.B. andere Maschinen. Eine Rolle in diesem Zusammenhang kann aber auch ein Mensch sein, der verschiedene Funktionen ausfüllt und während der Laufzeit die Wissensbasis erweitert. Rollen besitzen ebenso wie Requisiten eindeutige Namen, die bei der Spezifikation der einzelnen Ereignisse benutzt werden.

2.4 Ereignisse

Die von einem Roboter auszuführende Aktion „Bewege Objekt von A nach B" ist eine relativ komplexe Handlung. So muß der Roboter das Objekt erst greifen, dann über einen noch zu bestimmenden Weg transportieren, und zum Schluß muß er es wieder loslassen. Der zentrale Teil eines Skripts enthält daher eine Verfeinerung dieser Aktion durch Angabe einer Folge von Ereignissen. Mehrere Ereignisfolgen unterscheiden sich häufig nur in bestimmten Attributen wie Zeit, Stellung und Art der bewegten Objekte. Das Skript beschreibt den Prototyp einer Handlung.

Ereignisse werden durch ein Verb (eine Handlung) charakterisiert. Wir definieren ein *Ereignis* in der folgenden Weise : Ein Ereignis ist eine Zustandsänderung, die zu einem definierten Zeitpunkt beginnt und eine gewisse Zeit dauert.

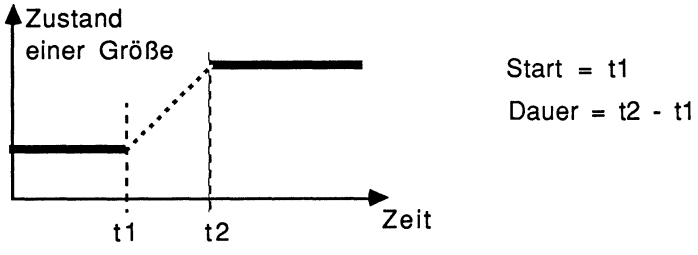


Bild 1: Ereignis

Das *Nullereignis* ist ein Ereignis, bei dem keine Zustandsänderung beobachtet wird. Die grafische Visualisierung eines Ereignisses zeigt Bild 10.

2.5 Zeiten

Die Angabe relativer und absoluter Zeiten erlaubt die zeitliche Einplanung von Aktionen, Synchronisierung von Aktionen und Fehlerbehandlung. Mit diesen Angaben sind Aussagen über die Gesamtzeit des Arbeitsvorganges möglich. Weiterhin kann bei bekannter Bahn hieraus auf die Bewegungsgeschwindigkeit geschlossen werden. Es erscheint sinnvoll, die zeitliche Planung auf der Ebene ganzer Skripte vorzunehmen, das Skript ist dann das Atom der zeitlichen Planung.Wir verlieren dadurch zwar die Möglichkeit, Ereignisse im Skript, die parallel ablaufen könnten, auch wirklich parallel ablaufen zu lassen, verringern jedoch die Komplexität der Zeitplanung. Sofern ausreichend leistungsfähige Planungsmodule vorliegen, können potentiell parallel ausführbare Skripte erkannt und nebenläufig abgearbeitet werden.

2.6 Weitere Eigenschaften

Zur Spezifikation von Ereignissen ist im allgemeinen die Kenntnis der Stellungen der an den Ereignissen beteiligten Objekte und Requisiten erforderlich. Diese Stellungen können wie üblich durch homogene Koordinaten beschrieben werden.

In vielen Anwendungen werden noch weitere physikalische Größen zur Spezifikation von Ereignissen benötigt, wie z.B. Kraft, Drehmoment oder Beschleunigung, die problemlos in das Konzept der Skripte passen. Wir wollen sie hier aber der Einfachheit halber nicht einführen.

3 Wissensgestützte Programmierung von Robotern

Der Skriptmechanismus stellt für uns nur einen *Teil* der wissensgestützten Programmierung von Robotern dar.

3.1 Erster Ansatz

Wir gehen vorläufig von folgender funktionaler Gliederung der wissensgestützten Programmierung von Robotern aus:

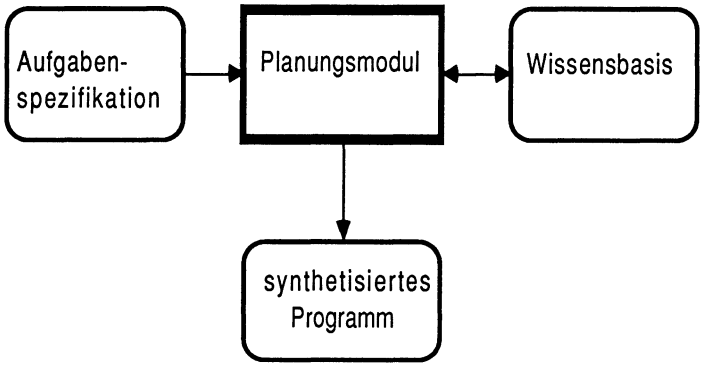


Bild 2: Wissensgestützte Programmierung von Robotern

Die Spezifikation der Aufgabe, die der Roboter lösen soll, kann durch Relationen beschrieben werden. Wir stellen uns Relationen vor, wie sie in ähnlicher Form auch in RAPT [Popp 78] enthalten sind, also z.B. :

Teil1 auf Teil2
Schraube1 in Loch1

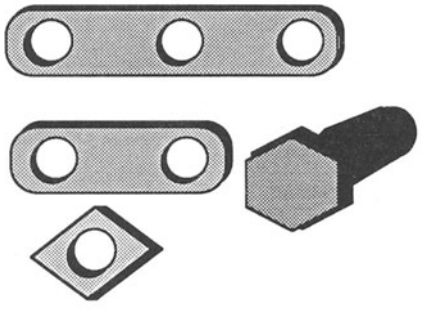
Diese Relationen werden bei komplexeren Aufgaben zu semantischen Netzen verknüpft.

Die meisten Planungsmodule im Bereich der künstlichen Intelligenz werden durch Produktionensysteme realisiert. Bevorzugt werden vor allem rückwärtsverkettete Produktionen, da sie die hierarchische Planung vereinfachen. Erfolgt die Planung in Echtzeit, werden auch vorwärtsverkette Produktionsregeln benötigt, um Ausnahmesituationen behandeln zu können.

Die Wissensbasis enthält neben der Umweltbeschreibung des Roboters (die mit Hilfe von Rahmen beschrieben wird) und der Produktionenregeln auch Lösungsstrategien (Metaregeln), die die Auswahl der Produktionenregeln steuern.

3.2 Beispiel

Wir wollen die Leistungsfähigkeit unseres Modells der wissensgestützten Programmierung von Robotern an einem Beispiel zeigen : Der Roboter soll mit dem Kinderspielzeug „Baufix™" verschiedene Modelle auf - bauen. Eine Teilaufgabe ist die Verbindung zweier Teile mit Schrauben. Wir haben folgende Einzelteile:



Rahmen	Mutter
Fach	Inhalt
Instanzname	mu1
Gewindegröße	m10
Art	raute
Farbe	rot
Ort	[[3, 1, 4], [90, 0, 33]]

Bild 3: Zusammenzufügende Teile

Bei den Teilen handelt es sich um eine Lochstange mit drei Löchern, eine solche mit zwei Löchern, eine Sechskantschraube und um eine rautenförmige Mutter. Die einzelnen Eigenschaften eines Bauteils (Größe, Anzahl der Löcher, Stellung u.a.) wird in der Wissenbasis durch Rahmen dargestellt. Hier wird auch dargestellt, welche Operationen in unserer Spielzeugwelt möglich sind. Es stehen damit alle Informationen zur Verfügung, die der Planungsmodul benötigt. Die Aufgabenspezifikation erfolgt mit Hilfe einer Zeichnung oder den hierzu äquivalenten Relationen.

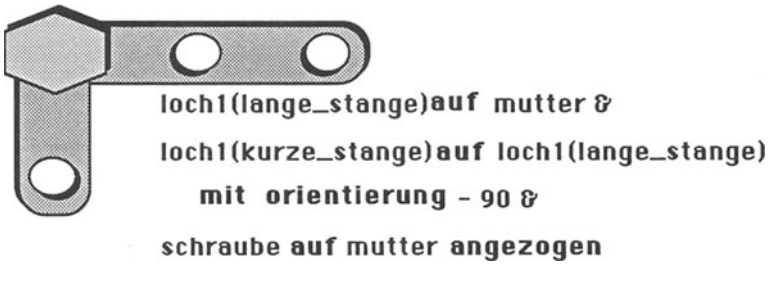


Bild 4: Beschreibung der Aufgabe

Der Planungsmodul erzeugt nun eine Reihe von Aktionen, die der Roboter ausführen soll. In unserem Fall könnte folgende Sequenz erzeugt werden :

```
lege Mutter1 von Lager in Vorrichtung
lege Lochplatte1 mit Loch1 über Mutter1
lege Lochplatte2 mit Loch1 über Lochplatte1 mit Orientierung 90°
lege Schraube1 auf Mutter1
verschraube Schraube1 mit Mutter1
```

Bild 5: Erzeugtes Programm

3.3 Skripte in der wissensgestützen Programmierung

Das erzeugte Programm stellen wir in mehreren Skripten dar. In den uns bisher bekannten Arbeiten wird spätestens hier zur prozeduralen Beschreibung übergegangen. Um weitere Probleme der wissensgestützten Programmierung wie Kollisionsvermeidung, Griffplanung und Überwachung der Planung durch Sensorik auf der Ebene der Wissensrepräsentation besser integrieren zu können, schlagen wir die Anwendung von Skripten vor. Diese Skripte werden durch einen weiteren Modul interpretiert.

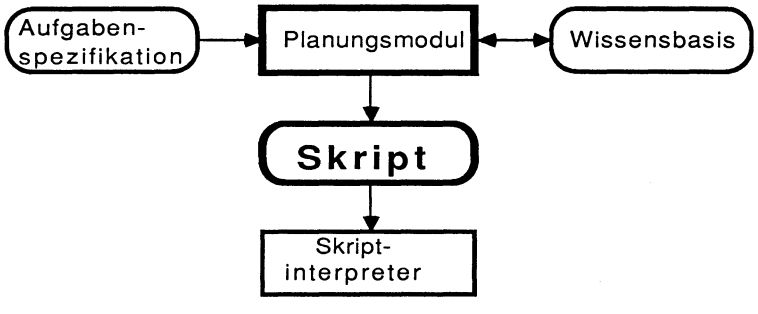


Bild 6: Wissensgestützte Programmierung mit Skripten

Ein einzelnes Skript beschreibt nun eine Aktion wie z.B.

lege Teil1 über Mutter1

Dieses Skript wird vom Planungsmodul ausgewählt, weil genau diese Aktion für die Aufgabenstellung notwendig ist. Eine grafische Darstellung eines Skriptes zeigt Bild 9.

3.4 Auswahl eines Skriptes

Es existieren immer mehrere Skripte, wobei das Planungsmodul aus der Anzahl der existierenden eines aussucht, das zur Erfüllung der spezifizierten Aufgabe geeignet scheint.

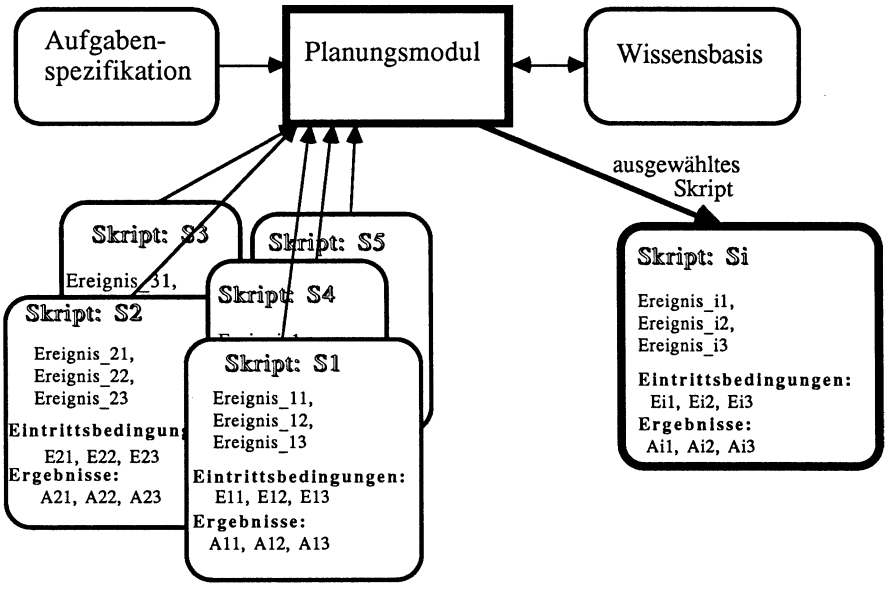


Bild 7: Auswahl eines Skriptes

Für diese Auswahl spielen verschiedene Kriterien eine Rolle. Die Auswahl wird aufgrund der Eigenschaften des Skriptes getroffen. Dabei gibt es Eigenschaften, die die Anwendung eines Skriptes nahelegen und andere, die notwendigerweise eingehalten werden müssen. So wird der Planungsmodul anhand verschiedener Eigenschaften des Skriptes eine Vorauswahl treffen. In einem Produktionensystem würden wir sagen : Es wird eine Konfliktmenge von Skripten gebildet. Anhand weiterer Bedingungen wird aus dieser genau ein Skript ausgewählt.

Ergebnisse

Ergebnisse des Skriptes sind eine geeignete Schnittstelle für einen rückwärtsverketteten Planungsmodul. Steht auf der linken Seite einer Produktionenregel ein Ziel, das mit dem Ergebnis eines Skriptes übereinstimmt, so sollte dieses Skript zur Auswahl erwogen werden. Bedingung dafür, daß es wirklich angewandt wird, ist die Erfüllung der Eintrittsbedingungen. Diese Zustandsbeschreibung (die Eintrittsbedingungen) bilden nun ein neues Unterziel eines hierarchisch arbeitenden Planungsmoduls.

Eintrittsbedingungen

So wie wir die Ergebnisse als Indiz für ein rückwärtsverkettetes Planungsmodul benutzen, bieten sich die Eintrittsbedingungen für eine vorwärtsgerichtete Planung an. Sie geht von einem gegebenen Zustand der Umwelt aus, der mit den Eintrittsbedingungen übereinstimmt und modifiziert die Umwelt entsprechend der im Skript spezifizierten Ereignisse. Das vom Skript nach seiner Ausführung erreichte Ergebnis kann neben den im Skript spezifizierten Requisiten ein Indiz für die Auswahl eines Skriptes bei einem gegebenen Umweltzustand sein.

Requisiten und Rollen

Wir haben gesehen, wie Skripte aufgrund von Ergebnissen und Eintrittsbedingungen vom Planungsmodul ausgewählt werden. In diesem Fall waren die an das Skript gebundenen Zustände Ursache der Aktivierung des Skriptes. Weitere Anhaltspunkte für die Auswahl eines Skriptes aus einer Konfliktmenge können die beteiligten Requisiten sein. Ist bekannt, daß ein bestimmtes Requisit benötigt wird, so liefert dies einen zusätzlichen Anhaltspunkt für den Planungsmodul, dieses Skript auszuwählen. Wollen wir unsere Verschraubungsaufgabe lösen, wird das Planungsmodul nur Skripte untersuchen, die das Requisit *Schraube* beinhalten.

Ereignisse

Namen von Ereignissen können, ebenso wie die Eigenschaften des Skriptes, Stichwort für die Aktivierung eines Skriptes sein. Der Planungsmodul kann also aufgrund von einzelnen Ereignissen, die im Planungsprozeß notwendig sind, ein bestimmtes Skript auswählen.

4 Echtzeitplanung und Fehlerbehandlung

Der bisher vorgestellte Formalismus erlaubt die ereignisorientierte Darstellung von Wissen als Grundlage für die Bewegungsplanung von Robotern. Insbesondere kann die Planung der Aktion in Echtzeit, das heißt unmittelbar vor ihrer Ausführung, stattfinden. Dies hat den Vorteil, daß das Planungsystem flexibler auf Fehler bzw. auf unvorhergesehene Ereignisse, die während der Ausführungsphase des Planes auftreten, reagieren kann. Bei einem System, das das Bewegungsprogramm nicht in Echtzeit generiert, könnten zwar schon in der Planungsphase mögliche Fehlerkonstellationen berücksichtigt werden. Dies hätte jedoch zur Folge, daß das synthetisierte Programm größer als nötig und vor allem weniger flexibel würde.

Typischerweise würde ein Planungsprozeß wie folgt ablaufen : Zuerst sucht der Planungsmodul nach einem möglichen Weg, die gestellte Aufgabe zu lösen. Erkennt er dabei, daß er sich auf einem Irrweg befindet, versucht er (z.B. mittels Backtracking), andere Möglichkeiten zur Erreichung des Ziels zu finden. Ein solches Backtracking ist selbstverständlich nur während der Planungsphase zulässig.

Hat der Modul einen Weg gefunden und einen Plan (also eine Folge von Skript-Aufrufen) aufgestellt, wird dieser Plan Aktion für Aktion ausgeführt. Tritt während der Planausführung ein Fehler auf, wird die Planausführung unterbrochen. Die Aufgabe des Planungsmoduls besteht nun darin, den Fehler zu beheben. Dabei ist aufgrund des veränderten Umweltzustandes eine erneute Planung erforderlich. Die dabei generierte Folge von Aktionen erzeugt entweder einen Zustand, der es erlaubt, daß der alte Plan fortgeführt wird, oder die Planung muß vollständig neu durchgeführt werden. Das Planungssystem muß deshalb mit zusätzlichen Informationen darüber ausgestattet werden, wann eine Aktion noch als „planmäßig erfüllt" angesehen wird, und in welchem Fall korrigierende Maßnahmen einzuleiten sind.

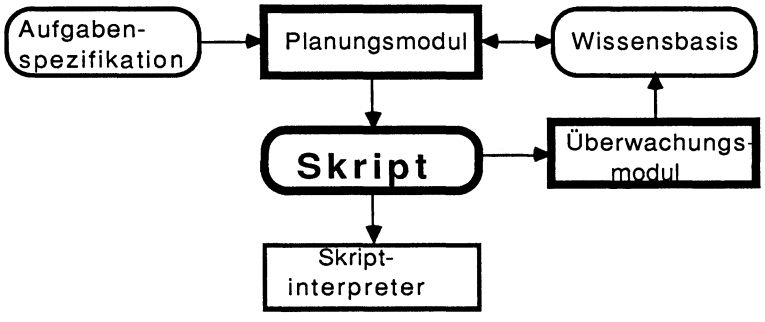


Bild 8: Skript mit Überwachung

Wir benötigen einen Mechanismus, der gewährleistet, daß ein vom Planungsmodul generiertes Programm wie vorgesehen auch dann ausgeführt wird, wenn zur Laufzeit Fehler auftreten. Unser Modell der wissens - gestützten Programmierung von Robotern sieht außerdem eine Wissenserweiterung während der Laufzeit vor. Tritt ein Fehler auf, für dessen Beseitigung dem Planungsmodul keine adäquaten Mittel zur Verfügung stehen, kann die Wissensbasis des Systems von einem menschlichen Experten so erweitert werden, daß bei erneutem Auftreten dieser Situation das System allein mit ihr fertig wird.

4.1 Fehlerdefinition

Wir sprechen immer dann von einem *Fehler*, wenn die Ausführung eines Ereignisses nicht in allen Phasen mit dem geplanten Ablauf des Ereignisses zusammenfällt.

Wir teilen die dabei auftretenden Fehler in die folgenden drei Kategorien ein :

I. : Fehler ohne Zwang zur Korrektur

II. : Fehler mit Zwang zur Korrektur, dabei Unterteilung in

a) Toleranzabweichungen

b) Logische Fehler

III. : Nicht vom System automatisch zu behebende Fehler

Unter Kategorie I. ordnen wir Fehler ein, deren Auftreten für den planmäßigen Abschluß der Aktion kein Hindernis darstellen. Beispiel : Eine Abweichung der vom Roboter gefahrenen Bahn von der ursprünglich geplanten Bahn ist im allgemeinen unproblematisch, wenn der eigentliche Zielpunkt dennoch positions- und zeitgenau angefahren wird, und wenn es nicht aus diesem Grund zu einer Kollision mit einem Hindernis kommt. Eine Maßnahme zur Korrektur der Bahn erübrigt sich in diesem Fall.

Ein Fehler nach Kategorie II a) oder II b) zwingt das System, Maßnahmen zur Fehlerbehandlung zu ergreifen. Wir sprechen von einem *logischen Fehler,* wenn während des Ablaufes der Aktion eine Umweltkonstellation auftritt, die im Plan nicht vorgesehen war. Beispiel : Der Roboter greift eine Schraube mit einem falschen Durchmesser, die sich unbeabsichtigt in der Zuführung befand. Es bestehen mehrere Möglichkeiten, die Aktion dennoch zu einem korrekten Ende zu führen. In jedem Falle aber wird der logische Ablauf der Aktion ein anderer sein, als der ursprünglich vorgesehene.

Wir sprechen von einem *Toleranzfehler,* wenn das Ergebnis der Ausführung der Aktionen des Plans zwar logisch richtig ist, jedoch nicht exakt mit dem eigentlich aufgrund der Planvorgaben erwarteten übereinstimmt. Dies wird der Regelfall sein, denn eine im mathematischen Sinne exakte Übereinstimmung zwischen erreichtem und gewünschten Ergebnis ist praktisch nicht erreichbar.

Tritt nach einer Aktion eine Abweichung auf, kann die Aktion durch eine weitere ergänzt werden, die versucht, den Toleranzfehler zu beheben. Ist die Abweichung jedoch gering, wird lediglich diese Abweichung in die Wissensbasis übernommen. Die tatsächlichen Ergebnisse der Aktion sind nun als Zustand der Umwelt gespeichert. In jedem Fall ist aber der eigentlich geplante Ablauf der Aktion nicht unterbrochen worden. Beispiel : Durch Mitaufnahme eines Fremdkörpers zu einem eigentlich zu greifenden Objekt positioniert der Roboter das Objekt an falscher Stelle. Der Positioniervorgang wird zum Zeitpunkt des Fehlerauftretens (nämlich des Einklemmens des Fremdkörpers) nicht unterbrochen, man kann sich jedoch vorstellen, daß die richtige Position durch einen Korrekturgriff erreicht wird.

Ein Fehler nach Kategorie III kann vom System nicht mit eigenen Mitteln behoben werden. Um die begonnene Aktion fortzuführen, ist die Mithilfe des Benutzers erforderlich. Insbesondere fallen in diese Kategorie alle Fehler, für die beim erstmaligen Auftreten noch kein adäquates Skript existiert. Wird ein solches vom Benutzer (interaktiv, direkt nach dem Auftritt des Fehlers) erzeugt, dann bleibt es im System gespeichert und steht beim Auftritt der nächsten vergleichbaren Fehlersituation zur Fehlerbehandlung zur Verfügung. Der Fehler rechnet nun nicht mehr zur Kategorie III, sondern zur Kategorie II.

4.2 Fehlererkennung

Es wurde schon erwähnt, daß die Bedingung für das Aufrufen eines Skriptes die Übereinstimmung des Umweltzustandes mit einem im Skript spezifizierten Eingangszustand ist. In der gleichen Weise haben wir einen Ausgangszustand definiert. Dem Planungsmodul ist also bekannt, welchen Zustand die Umwelt nach dem Verlassen des Skriptes (d.h. nach vollständiger Abarbeitung der Ereignisfolge) aufweist, sofern während des Ablaufs der Ereignisse kein Fehler aufgetreten ist. Der Vergleich des im Skript spezifizierten Ergebnisses mit dem tatsächlich erreichten Umweltzustand, sowie dessen Vergleich mit den Eintrittsbedingungen der aufzurufenden Skripte obliegt dem Überwachungsmodul (s. Bild 8). Dem Modul ist sowohl das momentan abgearbeitete Skript wie auch der Zustand der Umwelt bekannt. Er sorgt ebenso für eine ständige Aktualisierung der Wissensbasis in Abhängigkeit von der Umwelt. Fehler können durch Vergleich der nachgeführten Wissensbasis mit dem tatsächlichen Zustand der Umwelt entweder nach jedem Ereignis oder erst nach Beendigung des Skriptes ermittelt werden. Die Abprüfung des Umweltzustandes nach Ablauf eines jeden Ereignisses ist sicherlich aufwendiger als die Abprüfung der gesamten Ereignisfolge nach Verlassen des Skripts. Der Benutzer kann deshalb spezifizieren, welche Ereignisse das System nach ihrem Ablauf einzeln prüfen soll und welche nicht. Jede Ereignisbeschreibung enthält ein Fach, in dem der Benutzer diese Angabe macht.

Toleranzen

Zur Erkennung von Fehlern der Kategorie IIa) führen wir ein Fach für die Angabe einer generellen Toleranz in das Skript ein. In dieses Toleranzfach schreibt der Benutzer die maximal zulässige Toleranz für Zeiten, Wege und Kräfte (sowie möglicherweise andere physikalische Größen). Diese Angabe gilt für alle Ereignisse, falls der Benutzer nicht in die ebenfalls bei der Ereignisbeschreibung vorgesehenen Fächer jedes Einzelereignisses eine abweichende Eintragung macht.

4.3 Fehlerbehandlung

Asynchrone Ereignisse

Bislang haben wir nur synchron auftretende Ereignisse betrachtet, d.h. Fehler, die beim Ablauf eines Ereignisses bzw. des gesamten Skriptes erkannt werden konnten. Wir wollen nun noch zusätzlich Fehler zulassen, die während des Ablaufs des Ereignisses asynchron, d.h. zu einem unerwarteten Zeitpunkt und an unerwarteter Stelle auftreten können. Beispiel : Während Roboter 1 dabei ist, ein Teil zu greifen, verliert ein in der Nähe stehender Roboter 2 ein Teil. Das herunterfallende Teil schlägt Roboter 1 das gegriffene Teil aus dem Greifer. Wir geben dem Benutzer im Fach „Ausnahme" die Möglichkeit, Quellen für solche Ausnahmen zu spezifizieren. Diese Quellen werden für das ganze Skript nur einmal angegeben.

Abbruch eines Skriptes

Sobald ein Fehler der Kategorien II und III erkannt wird, wird die weitere Abarbeitung der Ereignisfolge abgebrochen. Der jetzt herrschende Zustand der Umwelt wird vom Überwachungsmodul in die Wissensbasis übertragen. Für den Planungsmodul ergibt sich nun folgende Zielsetzung : Erreicht werden muß auf jeden Fall ein Zustand, in dem die weitere Fortführung des Planes möglich ist, also ein Zustand, der dem intendierten Ergebnis des durch Auftritt des Fehlers abgebrochenen Skriptes möglichst nahekommt.

Für den weiteren Fortgang der Aktionen ist es nun entscheidend, ob das zuletzt bearbeitete Ereignis rücksetzbar war, oder nicht.

Bei *nicht rücksetzbaren* Skripten muß die von jetzt an ablaufende Ereignisfolge neu geplant werden. Unter Umständen ändert sich die gesamte Folge nach dem Auftritt des Fehlers. Zur Grundlage der nun ein - setzenden Neuplanung werden Skripte gemacht, die nur für Fehlersituationen geschrieben wurden. Diese Skripte unterscheiden sich jedoch sonst nicht von den für den normalen Ablauf vorgesehenen. Der Planungsmodul wird deshalb über eine Folge von Skripten versuchen, einen anderen Weg zur Erreichung des Ergebniszustandes zu finden.

Bei *rücksetzbaren* Skripten ist es dagegen nicht nötig, eine Neuplanung vorzunehmen, stattdessen wird die Änderung der Umwelt wieder vollständig zurückgenommen. Die dazu erforderliche Information ist aber im Skript bereits enthalten. Gelingt diese Rückführung, wird das Skript erneut ausgeführt, in der Hoffnung, daß der Fehler nicht nochmals auftritt.

Sofern ein Echtzeitplanungsystem vorliegt, kann von nun ab so vorgegangen werden, wie bei der Auswahl eines normalen Skripts : In Abhängigkeit vom Umweltzustand sucht der Planungsmodul ein Skript, dessen Eingangsbedingung seinen Aufruf ermöglicht.

5. Beispiel

Die folgenden beiden Bilder zeigen eine mögliche grafische Darstellung eines Skriptes und eines Ereignisses für einen Verschraubungsvorgang :

Name :	lege M von S nach Z	fahre nah zu S	
Einstellung:	lege Mutter von Lager in Aufnahme	öffne Hand	
Rollen :	Roboter = R	fahre zu S	
		greife M	
Requisiten:	Aufnahme = Z Mutter = M Lager = S	fahre nah zu Z	
		fahre zu Z	
Start:	13:30	öffne Hand	
		fahre weg	
Abweichungen:	5 % 5 % 5 %	Eintrittsbedingungen:	M in S M ist tragbar Z ist frei
Zeit: Wege Kräfte			
Ausnahmen:	Kollision ⇒ M in Hand S ist leer	Ergebnisse	M auf Z S ist leer Z ist belegt

Bild 9: Skriptbeschreibung in grafischer Darstellung

Ereignis:	greife M	
Rolle:	Roboter = R	
Objekt:	Mutter = M	
Stellung:	[[3, 4, 2], [90, 45, 90]]	
Dauer:	5 s	
Genauigkeit:	25 %	
Weg:	linear	
Abweichung:	Standard	
Ausnahme:	nichts gegriffen ⇒ Hand leer	
Soll Überwachung stattfinden		ja
Ist Ereignis rücksetzbar		ja

Bild 10: Ereignisbeschreibung

6 Zusammenfassung

Skripte stellen eines der von Hayes [Haye 79] geforderten Konzepte zur Beschreibung einer „naiven" Physik dar. Nach Kowalski [Kowa 86] ist zur Beschreibung von Ereignisabläufen, verbunden mit Aussagen, wann etwas wahr ist, die konzeptionelle Einbeziehung von Zeiten unabdingbar. In unserem Falle vereinheitlichen sich durch die Einbeziehung von Zeiten große Teile der wissensgestützen Programmierung von Robotern. Der Beschreibungsaufwand wird geringer als in herkömmlichen Systemen. Eine festdefinierte Wissensstruktur wie die vorgestellte erlaubt eine einfachere Kommunikation zwischen den Teilen eines großen Systems. Systemteile wie Kollisionsvermeidung, Planung und Fehlerüberwachung können getrennt voneinander entwickelt werden.

Zum gegenwärtigen Zeitpunkt befindet sich das System noch in der Konzeptionsphase. Es ist für die nächste Zukunft geplant, einen Prototyp unter Modula-Prolog auf einer VAXstation II zu implementieren.

Literatur

[Gini 83] Towards Automatic Error Recovery in Robot Programs,
Maria Gini, Giuseppina Gini,
in Proceedings of the 8th International Joint Conference on AI, Karlsruhe, 1983

[Haye 79] The Naive Physics Manifesto, Patrick J. Hayes,
in Expert Systems in Microelectronic Age, Donald Michie (Ed)
Edinburgh University Press, 1979

[Kowa 86] A Logic-based Calculus of Events
Robert Kowalski, Marek Sergot,
in New Computer Generation 4 (1986)

[Lamp 81] „Atomic Transaction" in Distributed Systems,
Architecture and Implementation
An Advanced Course in LNCS, Vol 105, Springer-Verlag 1981

[Niem 85] Semantische Netze als Ansatz zur Repräsentation von Wissen für die
automatische Bildanalyse, H. Niemann, G. Sagerer,
in Robotersysteme 1, 1985

[Popp 78] RAPT : A language for describing assemblies
Popplestone et al.
The Industrial Robot, Sept. 1978

[Scha 77] Scripts, Plans, Goals and Understanding,
Roger C. Schank, Robert P. Abelson,
Erlbaum, Hillsdale, N.Y., 1977

[Scha 81] Inside Computer Understanding,
Roger C. Schank, C.K. Kiesbeck (Eds),
Erlbaum, Hillsdale, N.Y., 1981