Erklärungen und Planungshilfen bei der funktionalen Programmierung

Claus Möbus, Heinz-Jürgen Thole, Olaf Schröder

Universität Oldenburg, Fachbereich Informatik, D - 26111 Oldenburg
Claus.Moebus@arbi.informatik.uni-oldenburg.de

Zusammenfassung

ABSYNT ist eine intelligente Problemlöseumgebung, die Personen bei der Bearbeitung von Programmieraufgaben in einer funktionalen, visuellen Sprache unterstützt. Der Benutzer testet Prüfhypothesen und erhält bei Bedarf Ergänzungsvorschläge. ABSYNT basiert auf einer Theorie des Wissenserwerbs und Problemlösesns und ermöglicht freies, exploratives Arbeiten.
Neben der Arbeit mit den Programmierkonstrukten kann der Benutzer auch seine Ziele und Intentionen zum Ausdruck bringen. Diese Komponente von ABSYNT beruht auf einem Transformationsansatz, welcher auch für die Ableitung von Erklärungen und weiterführenden Planungshilfen genutzt werden kann. Die Erweiterung von ABSYNT um Erklärungen und Planungshilfen wird gegenwärtig implementiert und ist Thema des vorliegenden Beitrags.

Einleitung

Intelligente Wissenskommunikationssysteme, wie Hilfesysteme, Tutorsysteme und Problemlöseumgebungen, unterstützen ihre Benutzer dabei, sich über das Lösen von Problemen oder allgemein die Bearbeitung von Aufgaben in einen Gegenstandsbereich einzuarbeiten. Für die Akzeptanz solcher Systeme ist es wesentlich, daß sie

- den Benutzer in seiner Freiheit zum Entwurf eigener Lösungsideen, also in seiner Kreativität, nicht zu sehr einengen
- dem Benutzer Informationen und Hilfen anbieten, die auf das aktuelle Problemlösegeschehen Bezug nehmen und möglichst auch den aktuellen Wissensstand des Benutzers berücksichtigen
- dem Benutzer gegenüber ihre Vorschläge, Hilfen usw. begründen können.

Die Entwicklung eines intelligenten Wissenskommunikationssystems ist ein aufwendiger Prozeß, der ohne eine Theorie des Wissenserwerbs und des Problemlösens nicht sinnvoll möglich ist. Aufbauend auf einer solchen Theorie haben wir u.a. ABSYNT (Abstract Syntax Trees) entwickelt, eine intelligente Problemlöseumgebung, die ihre Benutzer beim Erwerb von Programmierwissen in einer funktionalen graphischen Programmiersprache unterstützt. Die Kernannahmen unserer theoretischen Konzeption des Problemlösens und Wissenserwerbs sind (vgl. Möbus, 1991; Möbus, Schröder, Thole, 1992):

- Neues Wissen wird bevorzugt im Anschluß an Stocksituationen erworben. Stocksituationen führen zum Einsatz schwacher, wissensarmer Heuristiken (z.B. Fragen stellen, sich Informationen beschaffen usw.), die zur Überwindung der Stocksituation und damit zum Erwerb neuen Wissens führen (vgl. z.B. van Lehn, 1988).

• Bereits erworbenes Wissen wird durch wiederholte Anwendung optimiert (z.B. Wissenskompilation; Anderson, 1983).

• An Problemlöseprozessen können die Phasen des Abwägens zwischen verschiedenen Problemlösezielen, des Planens einer Lösung, der Ausführung des Plans und der Bewertung des Ergebnisses unterschieden werden (Gollwitzer, 1991; Heckhausen, 1989).

Für die Entwicklung eines Wissenskommunikationssystems bedeutet dies:

• Das System sollte Hilfen bereitstellen, aber den Benutzer nicht von sich aus unterbrechen, denn er wird von sich aus in Stocksituationen aktiv auf die Hilfen zugreifen.

• Hilfeinformation sollte an die jeweilige Problemlösephase und den jeweiligen Wissensstand des Benutzers anknüpfen, damit sie die aktuelle Stocksituation möglichst überwinden hilft und nicht weitere Stocksituationen provoziert.

Wir haben diese Kriterien in ABSYNT durch einen *Hypothesentestansatz* realisiert. Der Benutzer formuliert gegenüber dem System die Hypothese, daß sein Lösungsentwurf oder ein Teil seines Entwurfs richtig im Sinne der Aufgabenstellung ist. Das System gibt entsprechende Rückmeldung. Im nächsten Schritt kann sich der Benutzer bei weiterem Bedarf Ergänzungsvorschläge geben lassen (Möbus, Thole, 1990; Möbus, Schröder, Thole, 1992).
ABSYNT-Programme sind Bäume, deren Knoten primitive und selbstdefinierte Operatoren, Parameter und Konstanten sind. Darüber hinaus gibt es Zielknoten, mit denen der Benutzer noch nicht näher ausdifferenzierte Programmierziele repräsentieren kann. Die Zielknoten basieren auf dem im Münchner CIP-Projekt entwickelten Transformationsansatz (Bauer et al., 1987; Partsch, 1990), der die systematische, korrektheitserhaltende Ableitung funktionaler Programme aus Spezifikationen mittels Transformationsregeln erlaubt. Die ABSYNT-Zielknoten sind graphische Umsetzungen von Transformationsschritten (Möbus, Thole, Schröder, 1993; im Druck). Jeder Zielknoten steht für eine Spezifikation, die durch Doppelklick auf den Knoten sichtbar wird. Der Benutzer kann über Zielknoten Hypothesen testen und somit bereits in sehr frühen Lösungsstadien vom System Information darüber erhalten, ob er sich auf einem richtigen Lösungsweg befindet. Abb. 1 zeigt einen halbfertigen Entwurf zu der Aufgabe "DIFF BY DIFF 1": Es soll ein Programm konstruiert werden, das eine natürliche Zahl von einer Zahl abzieht, wobei der Subtraktionsoperator nur mit der "1" als zweitem Argument verwendet werden darf.
Das Programm wird konstruiert, indem mit der Maus aus der Werkzeugleiste am linken Bildrand Knoten aktiviert und im Editor positioniert werden. Neben dem primitiven Operatorknoten ">" enthält der Entwurf den selbstdefinierten Operatorknoten "DIFFDIFF1", die Parameter A und N sowie noch einige unbenannte Parameter, eine unbenannte Konstante, "Schatten", die für noch nicht realisierte Teilbäume stehen, sowie die Zielknoten "SUB1", "EQUAL0", "CASE" und "DIFF BY DIFF 1".

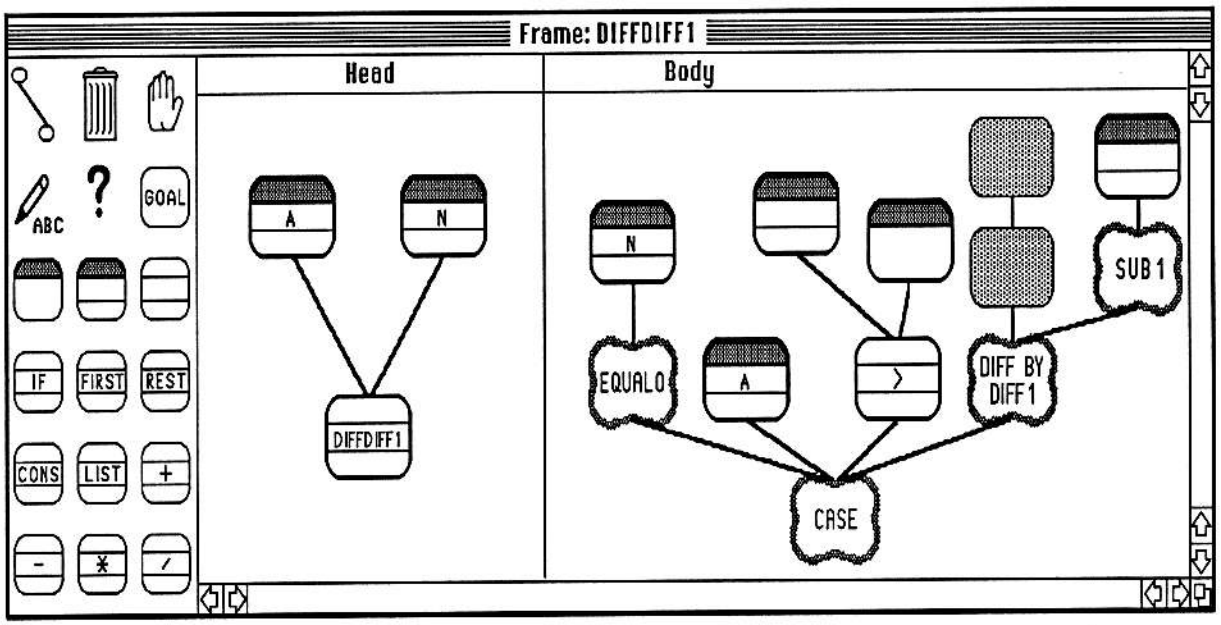

Abb. 1: Visueller Editor von ABSYNT mit einem unfertigen Lösungsentwurf

Im Sinne unserer Designkriterien für Wissenskommunikationssysteme stellt ABSYNT also Hilfen bereit; der *Benutzer* entscheidet, wann er Hypothesen testet und Ergänzungsvorschläge anfordert (Beispiele hierzu siehe unten). Das System erkennt für 42 Programmieraufgaben mehrere Millionen Lösungsentwürfe, so daß auch für ausgefallene Entwürfe Rückmeldung und Hilfe angeboten werden kann. Die hypothetischen Problemlösephasen des *Abwägens* und *Planens* werden durch die Zielknoten unterstützt, während die Operatorknoten der *Ausführungs*phase zuzuordnen sind. Das Hypothesentesten ist ein Werkzeug zur *Bewertung*. Außerdem steht ein visueller Trace zur Verfügung. Ein Benutzermodell, das auf den aktuellen Wissensstand des jeweiligen Benutzers zugeschnittene Ergänzungsvorschläge ermöglicht, ist teilweise realisiert.

In empirischen Untersuchungen mit ABSYNT haben sich vor allem zwei Aspekte als Kritikpunkt erwiesen:

- fehlende Erklärungen der Ergänzungsvorschläge
- fehlende Hilfen zur Ausdifferenzierung von Zielknoten. Es gibt z.B. keine Hinweise, wodurch der Zielknoten "DIFF BY DIFF 1" in Abb. 1 ersetzt werden kann.

Im folgenden werden wir an einem Beispiel darstellen, wie beide Aspekte auf der Basis des CIP-Transformationsansatzes in ABSYNT realisiert werden können.

Erklärungen und weiterführende Planungshilfen

Der CIP-Transformationsansatz kann sowohl für Erklärungen für die Ergänzungsvorschäge als auch für Hinweise zur Weiterentwicklung von Zielknoten genutzt werden. Ist eine Hypothese

positiv zurückgemeldet worden (d.h. sie kann vervollständigt werden), so können auf der Basis des CIP-Ansatzes folgende Hilfen und weiterführende Informationen gegeben werden:

• An offenen Kanten sowie an Schatten können Ergänzungsvorschläge gegeben werden, dies entspricht dem gegenwärtigen Implementationsstand.

• An jedem Zielknoten soll eine Erklärung für den Teilbaum gegeben werden können, dessen Wurzel dieser Zielknoten darstellt. Das System soll also dem Benutzer erklären können, wie ein Ergänzungsvorschlag zustandegekommen ist ("Wie"-Erklärungen im Sinne von Ellis, 1989; Hammond, 1984; Schurz, 1992). Außerdem kann für jeden Zielknoten der nächste Planungsschritt angegeben werden. Der Benutzer kann also Information darüber erhalten, wie der jeweilige Zielknoten weiterentwickelt werden kann: *wozu* führt dieser Zielknoten?

• Für Parameter- und Konstantenknoten können ebenfalls Erklärungen gegeben werden.

Im folgenden wird dies anhand einer Sequenz von Hypothesen für die ABSYNT-Programmieraufgabe "DIFF BY DIFF1" illustriert. Formal repräsentieren wird diese Aufgabenstellung mit "**that num** x: x = a - n". Gesucht ist die Zahl x, für die gilt, daß x = a - n. Die Restriktion "Subtraktion mit 1" kann für dieses Beispiel vernachlässigt werden.
Im Sinne des CIP-Ansatzes wird nun ein Funktionsschema gebildet:

funct DD1 (**num** a, **nat** n) **num**: **that num** x: x = a - n

Abb. 2 zeigt eine vom Benutzer formulierte Hypothese, die das System als auf dem Weg zu einer Lösung liegend erkannt hat. Dies wird dem Benutzer dadurch angezeigt, daß eine Kopie der Hypothese zurückgemeldet wird: Sie kann vervollständigt werden. Der Benutzer kann sich nun an dem Schatten und an der offenen Kante Ergänzungsvorschläge geben lassen. Das Ergebnis entspricht dann Abb. 3, einer graphischen Repräsentation des Funktionsschemas.
Als nächstes soll der Benutzer nun Erklärungen anfordern können. Die Erklärung für den vom System ergänzten "DIFF BY DIFF 1"-Knoten in Abb. 3 lautet, daß der Baum, dessen Wurzel dieser Knoten ist, die Aufgabenspezifikation darstellt. Die Parameter A und N sind die Parameter dieser Aufgabenspezifikation.
Außerdem soll der Benutzer an dem "DIFF BY DIFF 1"-Knoten Informationen über den nächsten Planungsschritt anfordern können. Die in der in Abb. 3 gezeigten Situation als nächstes anzuwendende CIP-Transformationsregel ist die Falleinführung. Tabelle 1 faßt die in Abb. 3 möglichen Erklärungen und Planungsschrittinformationen zusammen:

	Erklärung	nächster Planungsschritt
Zielknoten "DIFF BY DIFF 1"	Aufgabenspezifikation	Falleinführung
Parameter A	Parameter für Spezifikation	
Parameter N	Parameter für Spezifikation	

Tabelle 1: Erklärungen und nächste Planungsschritte für die Knoten des Körpers in Abb. 3

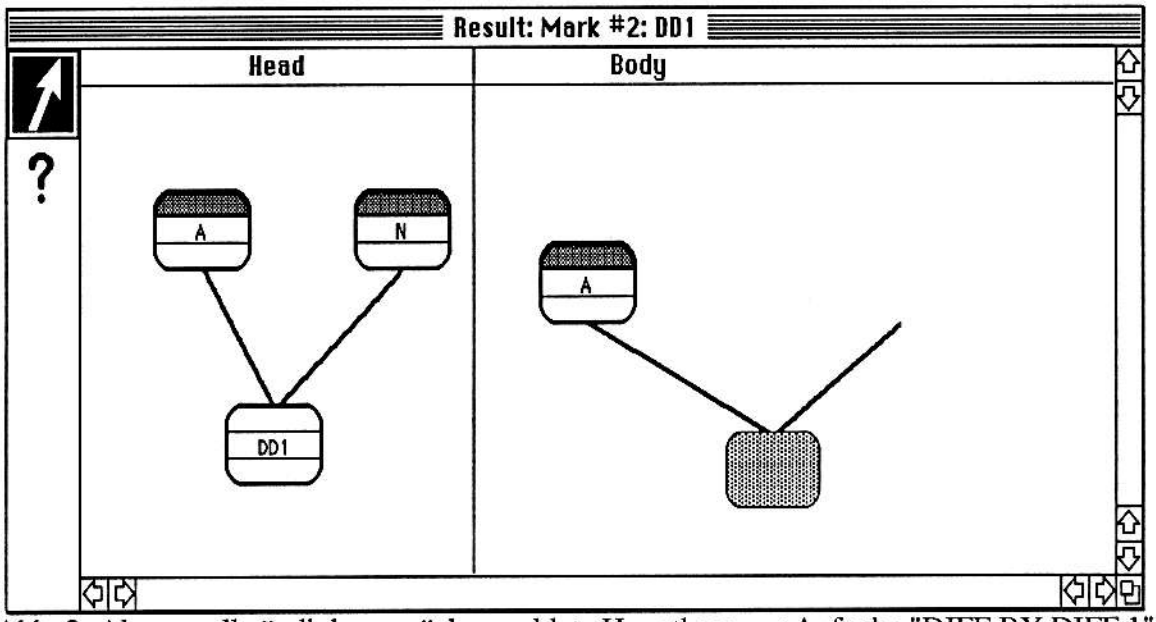

Abb. 2: Als vervollständigbar zurückgemeldete Hypothese zur Aufgabe "DIFF BY DIFF 1"

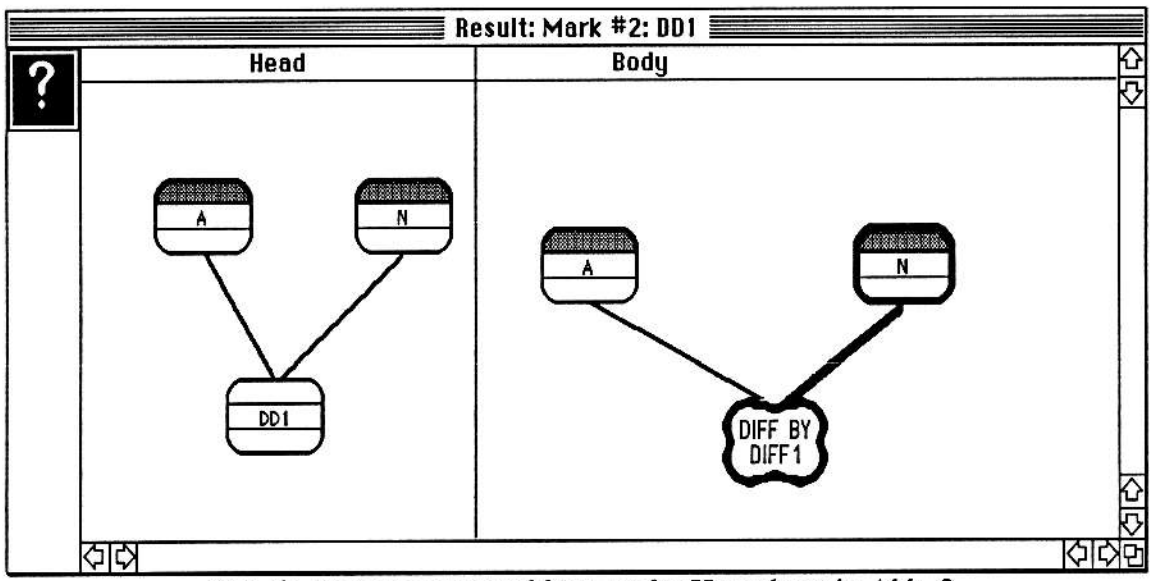

Abb. 3: Ergänzungsvorschläge zu der Hypothese in Abb. 2

Als nächstes, so nehmen wir an, ist die in Abb. 4 gezeigte Situation entstanden: Der Benutzer hat die Falleinführung durchgeführt ("CASE") und als erstes Prädikat das Ziel des Tests auf Gleichheit mit 0 ("EQUAL0") formuliert. Die entsprechende Hypothese ist positiv zurückgemeldet worden, wie in Abb. 4 dargestellt. Der Benutzer kann sich nun an den offenen Kanten Ergänzungen geben lassen (Abb. 5).

Die CIP-Regeln, die diese Ableitungsschritte beschreiben, sind die Falleinführung und die Prädikateinführung. Bei der Falleinführung werden z.B. zwei verschiedene Fälle unterschieden, die Aufgabenspezifikation unter diesen verschiedenen Fällen bleibt zunächst unverändert:

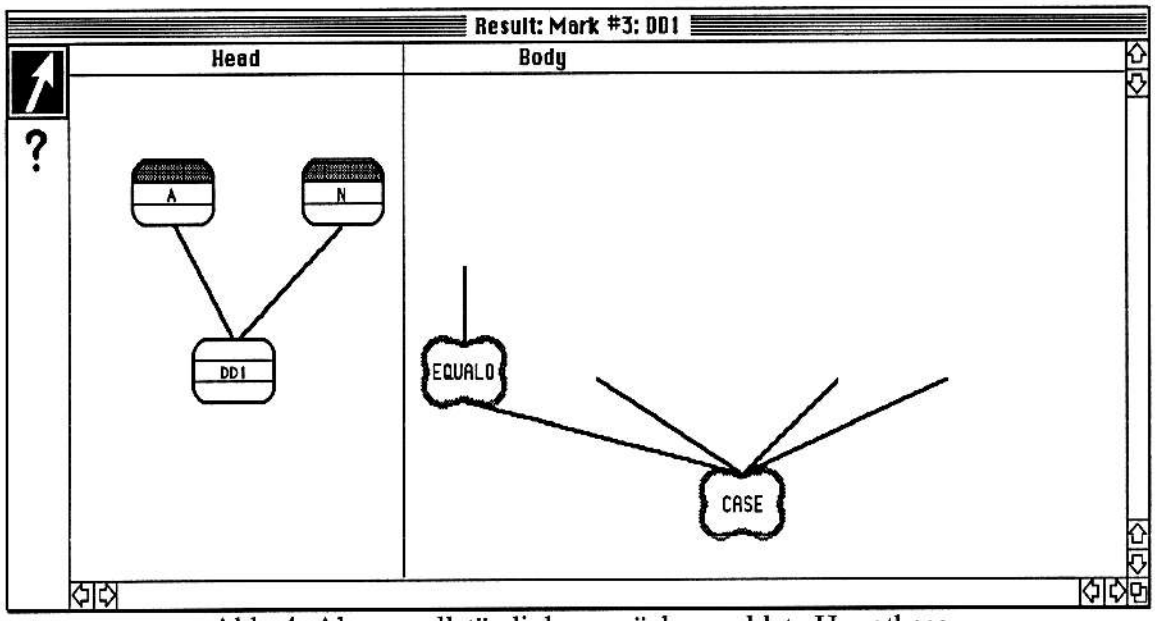

Abb. 4: Als vervollständigbar zurückgemeldete Hypothese

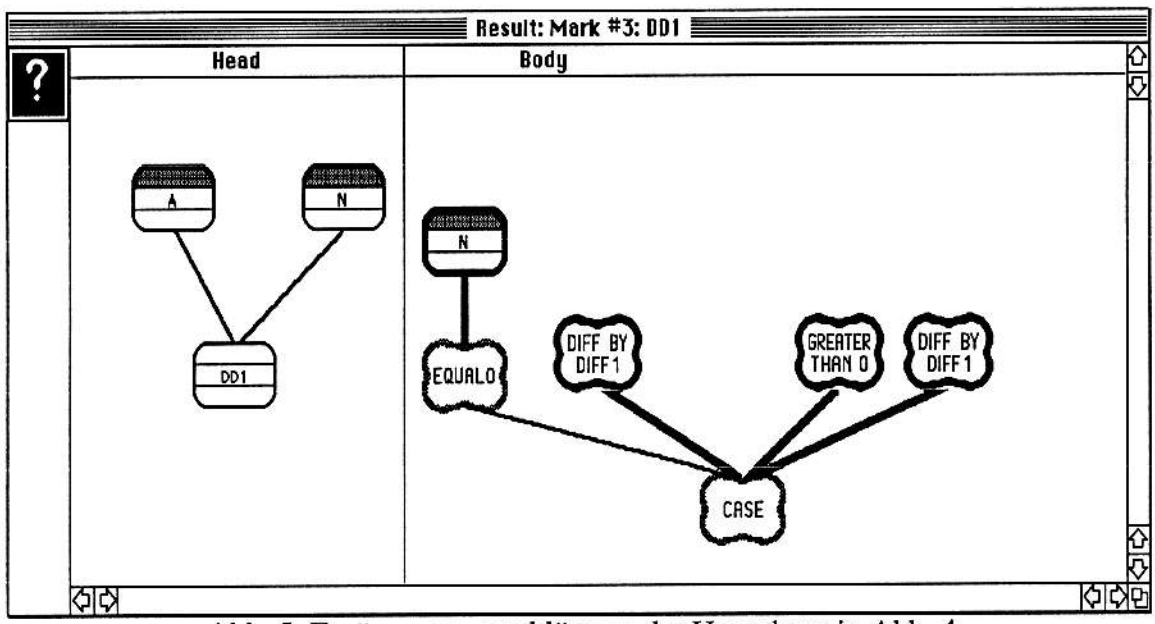

Abb. 5: Ergänzungsvorschläge zu der Hypothese in Abb. 4

funct DD1 (**num** a, **nat** n) **num**:
if B_1 **then that num** x: x = a - n
.....
if B_m **then that num** x: x = a - n (wobei m = 2)

Bei der Prädikateinführung wird der Argumentbereich eines Parameters zerlegt, z.B.

funct DD1 (**num** a, **nat** n) **num**:
if n = 0 **then that num** x: x = a - n
if n > 0 **then that num** x: x = a - n

Dabei müssen verschiedene Restriktionen (z.B. keine Überlappung und keine Lücken im Argumentbereich von n) beachtet werden. *Wie* der Argumentbereich zerlegt wird (hier n = 0, n

> 0), wird durch die CIP-Regeln nicht spezifiziert. Hier kommt zusätzliches Programmierwissen ins Spiel, das im Falle einer ausführlicheren Erklärung ebenfalls herangezogen werden müßte.

Im nächsten Transformationsschritt müssen unter den jeweiligen Prädikaten (hier n = 0, n > 0) Inferenzen für die Aufgabenspezifikationen durchgeführt werden. Der CASE-Knoten könnte bereits durch einen lauffähigen ABSYNT-Operatorknoten (z.B. "if-then-else") ersetzt werden. Ebenso könnten die Zielknoten EQUAL0 und GREATER_THAN_0 durch lauffähige Programmfragmente ersetzt werden. Die Erklärungen und weiterführenden Planungsschritte für die Zielknoten in Abb. 5 sehen also folgendermaßen aus (Tabelle 2):

	Erklärung	nächster Planungsschritt
Zielknoten CASE	Falleinführung	Ersetzung durch Teilbaum aus Operatoren
Zielknoten EQUAL0	Zerlegung des Argument-bereichs von N	Ersetzung durch Teilbaum aus Operatoren
Linker Zielknoten "DIFF BY DIFF 1"	Übernahme der Aufgaben-spezifikation unter EQUAL0	Bedingte Inferenz unter dem Prädikat EQUAL0: Vereinfachung der Spezifikation
Zielknoten GREATER_THAN_0	Zerlegung des Argument-bereichs von N	Ersetzung durch Teilbaum aus Operatoren
Rechter Zielknoten "DIFF BY DIFF 1"	Übernahme der Aufgabenspezifikation unter GREATER_THAN_0	Bedingte Inferenz unter dem Prädikat GREATER_THAN_0: Vereinfachung der Spez.
Parameter N	Parameter für Zerlegung des Argumentbereichs von N	

Tabelle 2: Erklärungen und nächste Planungsschritte für die Knoten des Körpers in Abb. 5

Abb. 6 zeigt einen weiteren Schritt auf dem Weg zu einer Lösung. (Die Edierhandlungen können durch Problemlösen oder durch Auswahl von Ergänzungsvorschlägen ausgeführt worden sein.) Die Inferenzen unter den Prädikaten sind teilweise durchgeführt worden. Während sich die Aufgabe unter dem ersten Prädikat zu dem Parameter A reduziert, ergibt sich unter dem zweiten Prädikat die Spezifikation einer Teilaufgabe (CIP-Regel "choice and quantification", vgl. Möbus, Thole, Schröder, 1993; im Druck). Im entsprechenden rechten Zweig des CASE-Knotens befindet sich in Abb. 6 noch eine Lücke, für die ein Ergänzungsvorschlag angefordert werden kann (Abb. 7). An dieser Stelle kann nun in einem weiterführenden Planungsschritt die Faltung durchgeführt werden, d.h. der Zielknoten "DIFF BY DIFF 1" kann durch den selbstdefinierten Operatorknoten "DD1" ersetzt werden (Rekursion). Tabelle 3 zeigt die Erklärungen und weiterführenden Planungsschritte für die

Zielknoten sowie für einige Parameter in Abb. 7, die im Vergleich zu Abb. 5 neu hinzugekommen sind:

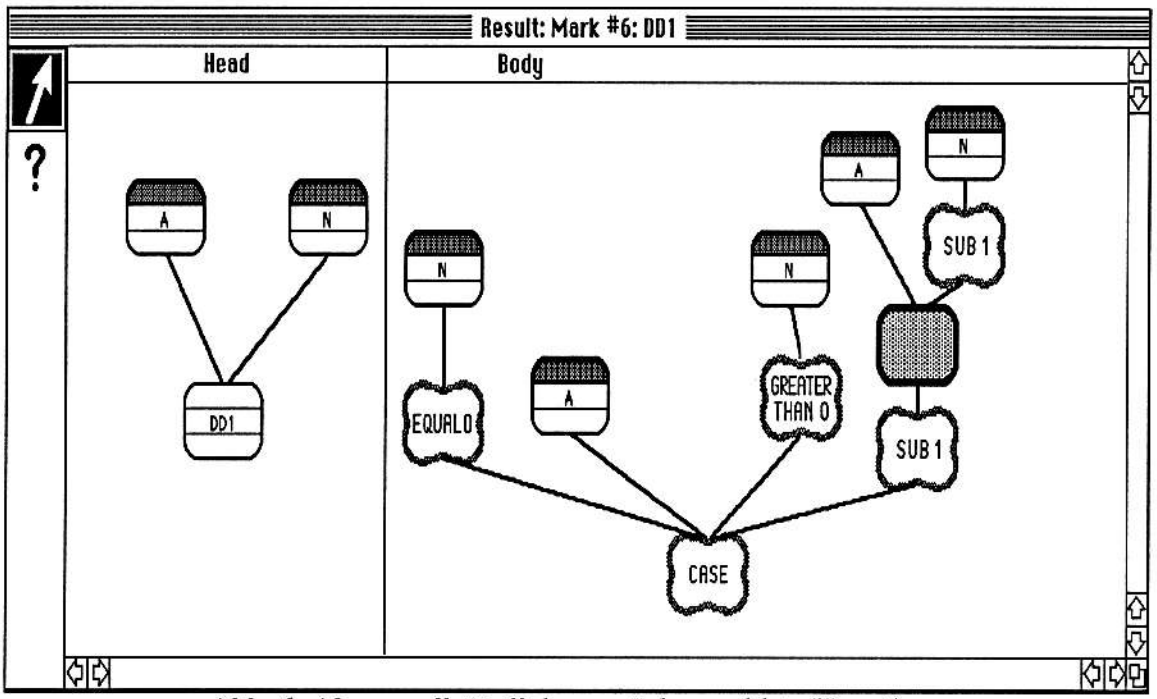

Abb. 6: Als vervollständigbar zurückgemeldete Hypothese

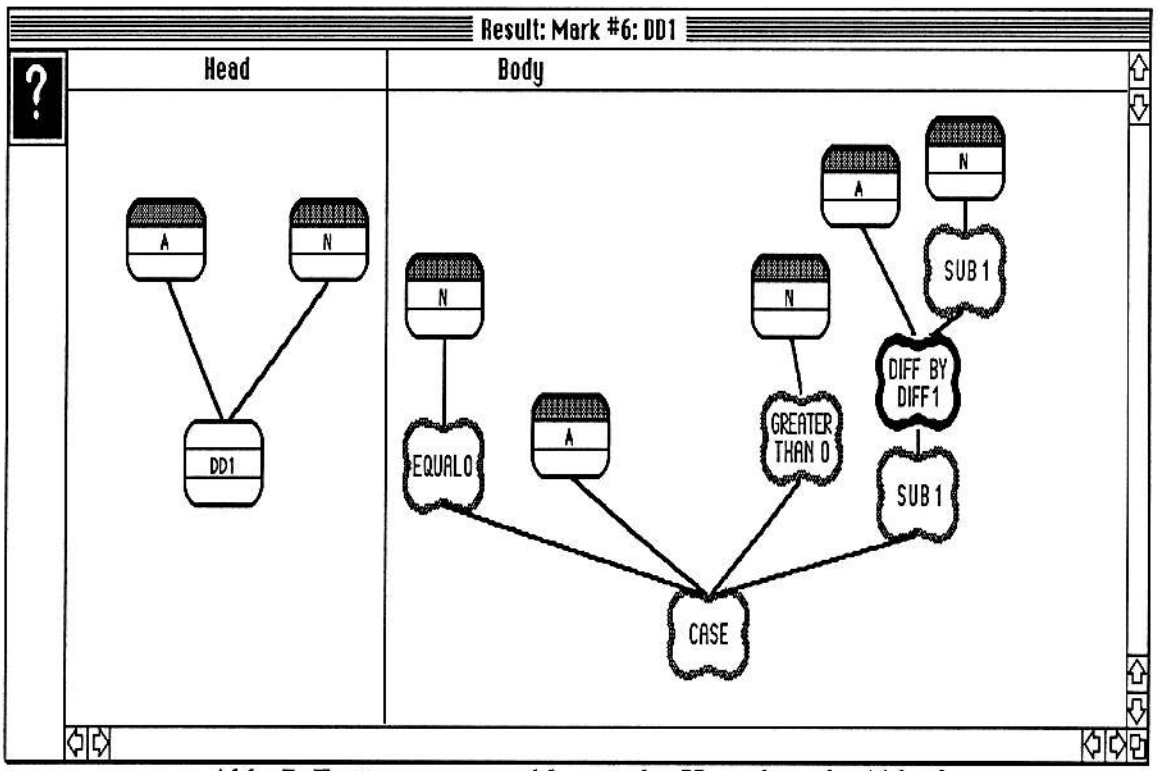

Abb. 7: Ergänzungsvorschlag zu der Hypothese in Abb. 6

Nach Durchführung des Faltungsschritts verbleiben nur noch Ersetzungen von Zielknoten durch lauffähige Operatoren bzw. Teilbäume, die in diesem Beispiel sehr einfach sind.

	Erklärung	nächster Planungsschritt
Parameter A am zweiten Zweig von CASE	Bedingte Inferenz unter dem Prädikat EQUAL0: Vereinfachung der Spezifikation	
Unterer Zielknoten SUB1	Bed. Inferenz unter dem Prädikat GREATER_THAN_0: Vereinfachung der Spezifikation	Ersetzung durch Teilbaum aus Operatoren
Zielknoten "DIFF BY DIFF 1"	Spezifikation einer Teilaufgabe, auf deren Lösung das Ziel SUB1 angewendet werden muß (= Lösung der Aufgabe unter der Bedingung GREATER_THAN_0)	Faltung
Oberer Zielknoten SUB1	Verringerung von N um 1 (N wird in Richtung auf die Stopbedingung EQUAL0 verringert.)	Ersetzung durch Teilbaum aus Operatoren
Parameter N von SUB1	Parameter für Verringerung von N um 1	
Parameter A von DIFF BY DIFF 1	Parameter für die Spezifikation einer Teilaufgabe	

Tabelle 3: Erklärungen und nächste Planungsschritte für Knoten des Körpers in Abb. 7

Ausblick

Neben Zielknoten, Parametern und Konstanten können auch Ergänzungen primitiver Operatorknoten erklärt werden. Dazu muß der Knoten (oder der Teilbaum, in dem sich dieser Knoten befindet) auf einen Zielknoten bezogen werden. Die Erklärung dieses Zielknotens kann dann wie oben dargestellt erfolgen.

Die skizzierte Komponente zur Generierung von Erklärungen für Ergänzungsvorschläge sowie von weiterführenden Planungshilfen zur Ausdifferenzierung von Zielknoten basiert auf dem Trace der Anwendung von Transformationsregeln. Insofern handelt es sich um einen "trace-basierten" Ansatz. Tracebasierte Ansätze sind z.T. kritisiert worden, weil die mit ihnen abgeleiteten Erklärungen oft sehr aufgabenspezifisch sind und damit selbst erklärungsbedürftig scheinen, z.B. durch "tieferliegende" Prinzipien (Chandrasekaran et al., 1989; Swartout, 1983; Wick & Thompson, 1992). Diese Kritik trifft jedoch u.E. auf den CIP-Transformationsansatz nicht zu, weil hier Wissen zur Ableitung einer Lösung vom aufgabenspezifischen Wissen getrennt ist. Es handelt sich hierbei um grundlegende Programmierprinzipien. Wie bereits erwähnt, legen die CIP-Regeln jedoch nicht alle Entwurfsentscheidungen fest (z.B. Wahl der Prädikate). Das hierfür benötigte zusätzliche Programmierwissen ist in dem hier beschriebenen Erklärungsansatz noch nicht berücksichtigt. Dies soll in einem späteren Schritt geschehen.

Der Erklärungsansatz wird gegenwärtig implementiert. Im Anschluß daran sind empirische Untersuchungen zur Wirksamkeit und Akzeptanz dieser Erweiterung des Systems vorgesehen.

Insgesamt hat sich der Transformationsansatz als ein sehr mächtiges Werkzeug erwiesen, das nicht nur eine Grundlage für das Arbeiten mit Zielknoten und die Diagnose von Intentionen darstellt, sondern auch die Ableitung von Erklärungen und weiteren Planungshilfen ermöglicht.

Literatur

Anderson, J.R., The Architecture of Cognition. Cambridge: Harvard University Press, 1983

Bauer, F.L., Ehler, H., Horsch, A., Möller, B., Partsch, H., Paukner, O., Pepper, P., The Munich Project CIP, Vol. II: The Program Transformation System CIP-S, Lecture Notes in Computer Science, Vol. 292, Berlin: Springer, 1987

Chandrasekaran, B., Tanner, M.C., Josephson, J.R., Explaining Control Strategies in Problem Solving, IEEE Expert, 1989, 9-24

Ellis, C., Explanation in Intelligent Systems, in C. Ellis (ed), Expert Knowledge and Explanation - The Knowledge-Language Interface, Chichester: Ellis Horwood Ltd., 1989, 108-156

Gollwitzer, P.M., Abwägen und Planen, Göttingen, Toronto: Verlag für Psychologie, 1991

Hammond, P., micro-PROLOG for Expert Systems, in K.L. Clark, F.G. McCabe (eds), micro-PROLOG: Programming in Logic, Englewood Cliffs: Prentice Hall, 1984, 294-319

Heckhausen, H., Motivation und Handeln, Berlin: Springer, 1989 (2. Aufl.)

Möbus, C., Wissenserwerb mit kooperativen Systemen, in: P. Gorny (Hg), Informatik und Schule 1991, Informatik: Wege zur Vielfalt beim Lehren und Lernen, GI-Fachtagung, Oldenburg, Okt. 1991, Proceedings, Informatik-Fachberichte 292, Berlin: Springer, 1991, 288 - 298

Möbus, C., Schröder, O., Thole, H.-J., A Model of the Acquisition and Improvement of Domain Knowledge for Functional Programming, J. of Artificial Intelligence in Education, 1992, 3(4), 449 - 476

Möbus, C., Thole, H.-J., Interactive Support for Planning Visual Programs in the Problem Solving Monitor ABSYNT: Giving Feedback to User Hypotheses on the Basis of a Goals-Means-Relation, in: D.H. Norrie, H.-W. Six (eds), Computer Assisted Learning. Proceedings of the 3rd International Conference on Computer-Assisted Learning ICCAL 90, Hagen, F.R.Germany, Lecture Notes in Computer Science, Vol. 438, Berlin: Springer, 1990, 36-49

Möbus, C., Thole, H.-J., Schröder, O., Interactive Support of Planning in a Functional, Visual Programming Language, in P. Brna, St. Ohlsson, H. Pain (eds), Proceedings of the 5th Int. Conf. on Artificial Intelligence in Education AIED 93, 362-369, 1993

Möbus, C., Thole, H.-J., Schröder, O., Diagnosis of Intentions and Interactive Support of Planning in a Functional, Visual Programming Language, erscheint in Proceedings of the NATO Advanced Research Workshop "The Use of Computer Models for Explication, Analysis, and Experiential Learning", Bonas, France, 12.-14.10.92, im Druck

Partsch, H.A., Specification and Transformation of Programs: A Formal Approach to Software Development, Berlin: Springer, 1990

Schurz, G., Erklärungsmodelle in der Wissenschaftstheorie und in der Künstlichen Intelligenz, in H. Stoyan (Hg.), Erklärung im Gespräch - Erklärung im Mensch-Maschine-Dialog, Berlin: Springer, 1992 (Informatik-Fachberichte 310), 1-42

Swartout, W.R., XPLAIN: a System for Creating and Explaining Expert Consulting Programs, Artificial Intelligence, 1983, 21, 285-325

Van Lehn, K., Toward a Theory of Impasse-Driven Learning. In: Mandl, H.; Lesgold, A. (eds): Learning Issues for Intelligent Tutoring Systems. New York: Springer, 1988, 19-41

Wick, M.R., Thompson, W.B., Reconstructive Expert System Explanation, Artificial Intelligence, 1992, 54, 33-70