Statistische Vorhersage von Benutzerverhalten und Erklärung von Hilfen in Petri Help

Knut Pitschke, Olaf Schröder, Claus Möbus

Universität Oldenburg
Fachbereich Informatik
Abteilung Lehr-Lernsysteme
Postfach 2503
26111 Oldenburg
{K.Pitschke, Schroeder, Moebus}@informatik.uni-oldenburg.de

Abstract

Diese Arbeit präsentiert einen Ansatz zur Vorhersage von Benutzerverhalten in Systemen, in denen kein explizites Domänenwissen verfügbar ist, sie zeigt Möglichkeiten auf, daraus Hilfen für den Benutzer abzuleiten und diese zu erklären. Das System benötigt ein Orakel (beispielsweise eine Verifikationskomponente), um das Verhalten der Benutzer a posteriori zu klassifizieren. Obgleich die Idee ursprünglich für den Einsatz in Intelligenten Tutoriellen Systemen entwickelt wurde, scheint sie auch in anderen Bereichen (z.B. Planen/Planerkennung) anwendbar zu sein.

Das System besteht aus einer Lernkomponente, die aus klassifizierten Benutzeraktionen Suchräume aufbaut, einer Vorhersagekomponente, die Wahrscheinlichkeitsaussagen über die nächsten Ziele und Aktionen des Benutzers macht und einer Adaptionskomponente, die den Vorhersagemechanismus kalibriert und ihn an den momentanen Benutzer anpaßt.

1. Einleitung

Intelligente Tutorielle Systeme (ITS) sind wissensbasierte Systeme, die den Lerner durch Hilfen, Erklärungen, Aufgaben und Anmerkungen in kompetenter und behutsamer Weise unterstützen sollen. Um diesen Ansprüchen gerecht zu werden, müssen sie sich an den individuellen Lernverlauf eines jeden Studenten anpassen können. Dies setzt ein Benutzermodell voraus, im Kontext von ITS wird es als Student Model bezeichnet [Kass 89; McCalla, Greer 92; Greer, McCalla 94]. Die Benutzung eines Student Models zur Bereitstellung adäquater Hilfen gliedert sich in Planerkennung und Planen. Unter Planerkennung verstehen wir das Erschließen eines intendierten Zieles aus manifesten Benutzeraktionen [Koehn, Greer 94]. Planung im Zusammenhang mit ITS wird zur Erzeugung von Hilfen, Hinweisen oder Vorschlägen benötigt. Dabei müssen dem Lerner Informationen angeboten werden, die sich auf den momentanen Zustand des Problemlöseprozesses beziehen und ihm helfen, sich in Richtung der intendierten Lösung zu bewegen. Die Benutzung von Ansätzen aus dem Bereich der Planerkennung im Kontext von ITS (z.B. [Bauer et. al. 91]) setzt in der Regel Domänenwissen über die Planungs- bzw. Lehrdomäne voraus [Kautz 91].

In dieser Arbeit wird ein Ansatz zur Verhaltensvorhersage von Benutzern solcher ITS vorgestellt, die über kein vorgegebenes Domänenwissen verfügen, aber stattdessen ein Orakel (im Sinne von [Valiant 84]) benutzen, um die Benutzerlösungen zu gegebenen Aufgaben zu bewerten.

2. Domänenwissen bei Benutzermodellierung und Planen

Ein Student Model beschreibt normalerweise das angenommene oder erschlossene Wissen eines Schülers in Relation zum Expertenwissen [Kass 89]. Das Benutzerwissen wird aus dem Verhalten des Schülers während des Dialoges mit dem System erschlossen. Die Interpretation seiner Antworten und Aktionen bezogen auf das vorhandene Expertenwissen ist Voraussetzung, um den aktuellen Status seines Wissens bezogen auf das Expertenwissen zu klassifizieren Diese Einordnung wird üblicherweise von einer Diagnosekomponente vorgenommen [Lusti 92], die mithilfe der vorhandenen Domänentheorie versucht, den Lösungsvorschlag des Studenten nachzuvollziehen [Ardissono et. al. 94]. Auf diese Weise ist es sowohl möglich, den Lösungsvorschlag auf Korrektheit zu prüfen (in Abhängigkeit davon, ob die gegebene Domänentheorie vollständig oder unvollständig ist), als auch, die vorhandene Teillösung in Richtung einer korrekten Lösung fortzuentwickeln [Möbus et al. 92b]. Wenn über die Plan- bzw. Lehrdomäne kein Expertenwissen verfügbar ist, ist es in der Regel weder möglich, das Planungsziel oder den Wissenszustand des Schülers zu diagnostizieren, noch zu planen oder einen Lösungsvorschlag für den Lernenden zu generieren. Im Bereich der Petri Netze, der Domäne unseres Systems Petri-Help [Möbus et. al. 93b], existiert jedoch keine konstruktive Designtheorie, die es gestatten würde, Netze aus temporallogischen Spezifikationen abzuleiten. Deswegen sind Standardverfahren aus dem Bereich des Planens oder der Benutzermodellierung nicht anwendbar.

3. Petri-Help

Petri-Help ist ein Hilfesystem, das Benutzer beim Erlernen von und bei der Modellierung mit Bedingungs-Ereignis Petri-Netzen unterstützt (Bild 2). Dem Lerner

wird ein Folge von Aufgaben aufsteigenden Schwierigkeitsgrades angeboten, für die er eine Lösung entwickeln soll. Während der Problemlösung kann der Student seine Lösungsvorschläge vom System überprüfen lassen oder Ergänzungsvorschläge anfordern. Die Bewertung von Lösungsentwürfen setzt sowohl eine formale Spezifikation der zu bearbeitenden Aufgabe als auch ein Orakel, das den Lösungsentwurf bezüglich dieser Spezifikation bewertet, voraus. In der Domäne der Petri-Netze ist es jedoch im allgemeinen unüblich, Aufgaben formal zu spezifizieren [Reisig 92]. Ausnahmen bilden [Olderog 91] und [Wedig 93], wo Petri-Netze aus Spezifikationen in Trace-Logik formal ableitbar sind.
In Petri-Help werden die zu lösenden Aufgaben durch eine Menge temporallogischer Formeln spezifiziert, welche Eigenschaften der Lösung beschreiben (Bild 1). Unsere Temporallogik ist eine, um die zeitlichen Prädikate "nexttime", "eventually" und "always" erweiterte Aussagenlogik, sie erlaubt branching-time und benutzt Step-Semantik. Diese Form der Spezifikation erlaubt es uns, Lösungsentwürfe oder Teile davon gegen die Spezifikation mittels Model-Checking [Josko 90] abzuprüfen. Dadurch läßt sich entscheiden, für welche Teile der Spezifikation das vorhandene Petri-Netz ein Modell ist bzw. welche Formeln erfüllt sind. Wegen der Beschränkung auf Bedingungs-Ereignis-Netze sind diese Modelle immer endlich und häufig zyklisch. Allerdings ist die Folge von Netzen (Netz-Entwurfsstadien), die ein Schüler im Zuge einer Problemlösung durchläuft, nichtmonoton in Bezug auf die in ihnen erfüllten Formeln.

Beschreibung der Stellen

Ws	Wirt schläft
Wba	Wirt ist bereit zur Auftragsannahme
Wbs	Wirt ist bereit zum Servieren
K	Küche hat Auftrag erhalten
Z	Zubereitung der Speisen
E	Essen ist fertig

Startbedingung: Ws

Temporallogische Formeln:

- $\Box(Wba \rightarrow \Diamond(Ws \wedge K))$
- $\Box(K \rightarrow \Diamond Z)$
- $\Box(Z \rightarrow \Diamond E)$
- $\Box(E \wedge Ws \rightarrow \Diamond Wbs)$
- $\Box(E \wedge Wba \rightarrow \Diamond Wbs)$
- $\Box(Wbs \rightarrow \Diamond Ws)$
- $\Box(Ws \rightarrow \Diamond Wba)$
- $\Box(\neg(Ws \wedge Wba))$
- $\Box(\neg(Ws \wedge Wbs))$
- $\Box(\neg(Wba \wedge Wbs))$
- $\Box(Ws \vee Wba \vee Wbs)$

□ bedeutet "always",
◊ bedeutet "eventually"

Bild 1: Temporallogische Beschreibung der "Restaurant"- Aufgabe

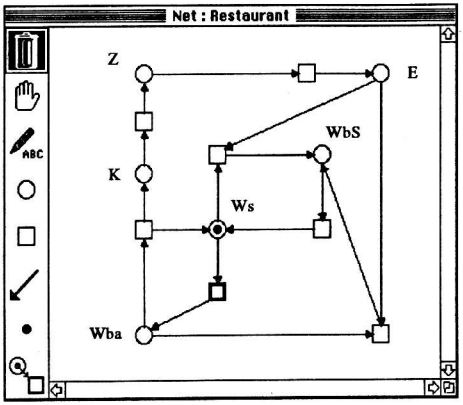

Bild 2: Eine Benutzerlösung zur "Restaurant" Aufgabe

Das bedeutet, daß eine Formel, die in einem früheren Stadium des Entwurfs als korrekt bewiesen wurde, zu einem späteren Zeitpunkt wieder falsch sein kann, obgleich dieser Schritt notwendig zu einer korrekten Lösung führt. Deswegen ist es nicht möglich, Netzfragmente, die einzelne Formeln erfüllen, zusammenzufügen und sicher zu sein, daß damit auch die Konjunktion der Formeln auf dem zusammengesetzten Netz gilt. Trotz dieser Nicht-Monotonie verändert sich die Menge der erfüllten Formeln im Laufe des Problemlöseprozesses: Anfangs sind gar keine Formeln erfüllt, am Ende (bei einer korrekten Lösung) gelten alle Formeln. Es muß also Zwischenzustände (Netzfragmente) auf dem Weg zu einer Lösung geben, in denen zusätzlich Formeln erfüllt wurden. Wir klassifizieren diese Zustände als "sicher" nach der folgenden induktiven Definition:
Der Anfangszustand gilt als "sicher". Jeder Zustand (jedes Netz), in dem eine echte Obermenge der im letzten "sicheren" Zustand erfüllten Formeln gilt, wird als "sicher" bezeichnet.
Aufgrund dieser Klassifikation von Benutzerlösungswegen werden vom System Designregeln gelernt, die als Basis für Ergänzungsvorschläge dienen.

Versuche mit Studenten haben gezeigt [Möbus et. al. 93b], daß die Möglichkeit, Netze auf Korrektheit überprüfen zu lassen, gerne und ausgiebig benutzt wird. Auch die Ergänzungsvorschläge des Systems, die auf Grund der gelernten Designregeln erzeugt wurden, wurden häufig in Anspruch genommen. Sinnvolle Ergänzungsvorschläge des Systems wurden jedoch erwartungsgemäß erst dann generiert, wenn Petri-Help adäquates Wissen aus früheren Problemlösesitzungen gelernt hatte. Kritisiert wurde, daß die angebotenen Ergänzungsvorschläge unzureichend auf den jeweiligen Benutzer abgestimmt waren. Oft wurde zu viel Information auf einmal präsentiert, oder aber, besonders in frühen Stadien der Problemlösung, wurde der Ratsuchende in eine Richtung gelenkt,

die er nicht intendierte. Dieses Problem tritt auf, weil noch kein Benutzermodell integriert ist, welches die Hilfegenerierung unterstützen könnte.

4. Ein adäquater Ansatz zur Benutzermodellierung

Wie bereits erwähnt, ist es in der Domäne der Petri-Netze nicht möglich, den Suchraum vorab aufzubauen. Stattdessen kann der Suchraum aus beobachteten und bewerteten Aktionen all derjenigen Benutzer gelernt werden, die mit dem System arbeiten. In diesem Zustandsraum werden spätere Benutzer aufgrund ihres beobachtbaren Verhaltens identifiziert und ihr erwartetes zukünftiges Verhalten ermittelt [Pitschke 93, 94]. Das bedeutet, daß eigentlich zwei Modelle vorliegen müssen: ein (theoretisches) Benutzermodell und ein (empirisches) Benutzungsmodell [Grunst et. al. 93]. Um Prognosen über das Benutzerverhalten aufzustellen, muß das Benutzermodell bezogen auf das Benutzungsmodell interpretiert werden.
In den folgenden Abschnitten wird beschrieben, wie die Zustandsräume durch Beobachtung des Benutzerverhaltens aufgebaut werden und wie neue Anwender in diesen Räumen identifiziert werden. Wir stellen wahrscheinlichkeitsorientierte Ansätze zur Vorhersage des Benutzerverhaltens vor und vergleichen diese. Dabei wird die Funktionsweise des Prognosemechanismus erläutert; insbesondere seine Adaptierbarkeit, die sich durch Vergleich zwischen früherer Vorhersage und tatsächlich beobachtetem Verhalten ergibt. Am Ende werden Einsatzszenarien zur Hilfegenerierung und zur Validierung von Designheuristiken aufgezeigt.

4.1 Repräsentation der beobachteten Ziele und Aktionen des Benutzers

Die Information über einen Benutzer beschränkt sich auf den Dialog zwischen ihm und dem System und seiner Interpretation bezogen auf den jeweiligen Kontext. Das System akquiriert Informationen auf zwei Ebenen: auf der Zielebene und auf der Aktionsebene. Als Ziel eines Benutzers wird das vollständige Lösen der gerade gewählten Aufgabe angesehen. Diese wird durch eine Menge temporallogischer Formeln spezifiziert. Eine mögliche Sequenz von Subzielen wäre jede Folge von Teilmengen der logischen Formeln, wobei jedes Teilziel eine echte Obermenge der im letzten Ziel erfüllten Formeln bezeichnet. Immer wenn ein Benutzer einen Netzentwurf gegen eine von ihm ausgewählte Menge der Spezifikationsformeln abprüft, wird angenommen, daß sein momentanes Subziel darin bestand, ein Netz zu konstruieren, das genau diesen Formeln genügt. Aus diesen Subzielen und den beobachteten Reihenfolgen ihres Auftretens wird ein gerichteter azyklischer Graph konstruiert, im weiteren als Zielgraph bezeichnet. Jede neue Beobachtung (d.h. Überprüfung einer Formelmenge durch den Benutzer) erweitert den Graphen.
Parallel dazu werden alle Aktionen der Benutzer aufgezeichnet. Diese werden in einem sog. Aktionsgraphen abgelegt. Die Knoten des Aktionsgraphen sind Petri-Netze, die Benutzer auf dem Weg zu ihrer Lösung entwickelten. Die Kanten entsprechen den Editieraktionen, die durchgeführt wurden, um ein Netz in das nächste zu überführen.
Ziel- und Aktionsgraph (Bild 3) sind verbunden. Jeder Zielknoten ist mit all denjenigen Netzen (Knoten) im Aktionsgraphen verbunden, die von Benutzern mit dem entsprechenden Subziel getestet wurden. Jeder Knoten im Ziel- und im Aktionsgraphen, der zumindest einen Nachfolger besitzt, wird um bedingte Übergangswahrscheinlichkeiten angereichert. Diese Wahrscheinlichkeiten sind die empirisch ermittelten relativen Häufigkeiten der jeweiligen Zustandsübergänge unter der Voraussetzung, daß eine bestimmte Vorgeschichte beobachtet wurde. Die bedingten Wahrscheinlichkeiten werden in jedem Knoten in folgender Form abgelegt:

$p_i(j \mid h)$ wobei:
i den aktuellen Knoten bezeichnet
j bezeichnet den Nachfolger
h steht für die beobachtete Vorgeschichte

Die Vorgeschichte besteht aus der Folge von Knoten, die ein Benutzer in dem jeweiligen Graphen durchlaufen hat. Diese bedingten Wahrscheinlichkeiten sind, zusammen mit einem Kriterienvektor, der für jeden Benutzer individuell erstellt und angepaßt wird, die Grundlage der Verhaltensvorhersage.

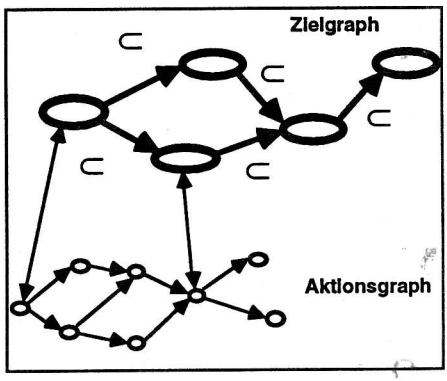

Bild 3: Ziel- u. Aktionsgraph repräsentieren das beobachtete Problemlöseverhalten einer Benutzergemeinde.

4.2 Formale Ansätze zur Verhaltensvorhersage

In diesem Abschnitt werden formale Modelle vorgestellt, um die aufgebauten Zustandsräume zu interpretieren und dadurch zu Vorhersagen über das Benutzerverhalten zu gelangen.

4.2.1 Stochastische Automaten

Stochastische Automaten (SA) [Buchanajew 94, Claus 71] unterscheiden sich von deterministischen bzw. nichtdeterministischen Automaten dadurch, daß für die Zustandsübergänge Wahrscheinlichkeitsfunktionen angegeben werden. Diese Funktionen beschreiben in jedem Zustand die Übergangswahrscheinlichkeiten bei gegebenen Eingaben. Als Eingabe kann im vorliegenden Fall die beobachtete Vorgeschichte des Benutzers angesehen werden. Für unsere Bedürfnisse müßte der Formalismus der SA in zwei Punkten erweitert werden:

- Die Übergangswahrscheinlichkeiten müssen dynamisch angepaßt werden (gemäß den Beobachtungen)
- Der Automat muß um neue (beobachtete) Zustände erweitert werden können.

4.2.2 Markov Modelle

Markov Modelle können als Sonderfälle Stochastischer Automaten angesehen werden [Claus 71]. Sie beschreiben Systeme mit abzählbaren Zustandsmengen [Wickens 82]. Die Übergangsfunktionen zwischen Zuständen sind probabilistischer Natur, d.h. es werden Wahrscheinlichkeiten angegeben, die den Übergang von einem gegebenen Zustand bei beobachteter Eingabe in einen Nachfolgezustand beschreiben. Desweiteren darf der Übergang, neben der Eingabe, ausschließlich vom aktuellen Zustand abhängen, eine Berücksichtigung der weiteren Vergangenheit, d.h. des Weges durch den Zustandsraum zum aktuellen Zustand findet nicht statt. Die komplette Vergangenheit muß demnach im aktuellen Zustand kodiert sein. Angewendet auf die gelernten Suchräume hieße dies, jeder Zustand wäre aufzuspalten in eine Menge von Zuständen, die sich nur durch den Weg, den der Benutzer einschlug um ihn zu erreichen, unterscheiden. Neben der Explosion der Anzahl der Zustände bringt diese Aufspaltung für uns einen weiteren Nachteil mit sich: Um Vorhersagen über das weitere Verhalten des Benutzers machen zu können, wird es nötig sein, über die Vergangenheit des einzelnen Benutzers zu abstrahieren, d.h. gezielt zu generalisieren. Bezogen auf ein Markov Modell hieße das, Identifikation des Benutzerzustandes in einer Äquivalenzklasse von Markov Zuständen. Generalisieren der Benutzergeschichte würde bedeuten, Auswahl einer geeigneten Teilmenge der Äquivalenzklassen, deren Vorgeschichten bestimmte Eigenschaften aufweisen.

4.3 Vorhersage des Verhaltens

Die Vorhersage des Benutzerverhaltens besteht in der Angabe der erwarteten Fortsetzung des bislang gezeigten Verhaltens bezogen auf das Verhalten aller zuvor beobachteten Benutzer.
Um in einem bekannten Zustand (des Benutzers) das nächste Ziel oder die nächste Aktion vorherzusagen, wird in dem entsprechenden Graphen nach gleichem Verhalten früherer Benutzer gesucht. Dazu werden die im aktuellen Knoten abgelegten bedingten Wahrscheinlichkeiten dahingehend untersucht, ob die beobachtete Vorgeschichte (siehe unten) des aktuellen Benutzers bei einer oder mehreren von ihnen als Vorbedingung auftritt. Ist dies der Fall, wird derjenige Nachfolgeknoten prognostiziert, der den höchsten Wahrscheinlichkeitswert aufweist. Tritt die Vorgeschichte nicht als Vorbedingung für einen Zustandsübergang im aktuellen Knoten auf, muß sie generalisiert werden, bis sie auf eine der Vorgeschichten matcht, die zum aktuellen Knoten führten. Die Kriterien, nach denen generalisiert wird, sind in einem benutzerspezifischen Kriterienvektor abgelegt. Die Kriterien werden benötigt, um aus der in Ziel- und Aktionsgraph abgelegten Information eine Vorgeschichte herauszufiltern, die derjenigen des momentanen Benutzers möglichst ähnlich ist. Das Ähnlichkeitsmaß wird dabei durch den Kriterienvektor und die Heuristik (siehe dort) bestimmt und unterliegt somit Änderungen. Die Filteroperation wird in zwei Schritten ausgeführt:
Zuerst wird die Vorgeschichte gemäß dem Kriterienvektor generalisiert. Führt dies zu keiner matchenden History, generalisiert die Heuristik den Kriterienvektor solange, bis dieser, angewendet auf die Vorgeschichte des Benutzers eine matchende History liefert. Danach entscheidet eine Konfliktlösungsstrategie, welche der möglichen Nachfolgeknoten am wahrscheinlichsten ist. Das Ergebnis des Vorhersageprozesses wird dann mit dem tatsächlich beobachteten Verhalten des Nutzers verglichen. Das Ergebnis diese Vergleiches fließt in die Adaption des Kriterienvektors und der Heuristik ein.

4.4 Der Kriterienvektor

Wie oben erwähnt, basiert der Vorhersageprozeß auf dem Kriterienvektor und dem Matchen einer passenden Vorgeschichte. Eine Vorgeschichte, die zu einem beliebigen Knoten führt, besteht zur Zeit aus der Folge von Knoten, die auf diesem Weg durchlaufen wurden und den dazu benötigten Zeitintervallen. Sie kann leicht erweitert werden um Information, die aus dem Dialog abgeleitet werden kann, wie z.B. Fehler des Benutzers oder Benutzung der Hilfeinformation.
Die Komponenten des Kriterienvektors geben an, inwieweit bei der Suche nach einer bekannten Vorgeschichte von derjenigen des Benutzers abgewichen werden darf. Der Kriterienvektor kann zum ersten Mal erzeugt werden, nachdem die ersten beiden Aktionen oder Ziele des Benutzers beobachtet wurden. Dadurch sind in dem ersten Knoten sowohl dessen Vorgeschichte als auch die Fortsetzung bekannt und es kann eine optimale Vorgeschichte zu dem ersten Knoten berechnet werden, die den zweiten Knoten prognostiziert hätte. Danach wird eine kleinste gemeinsame Generalisierung von dieser optimalen und der tatsächlichen Vorgeschichte berechnet. Der Kriterienvektor wird nun mit denjenigen Parametern initialisiert, die die Benutzervorgeschichte zu der gefundenen optimalen generalisieren würden.
Als Beispiel betrachten wir einen Schüler, der die Knoten a, b, c und d durchlaufen hat (egal ob im Ziel- oder im Aktionsgraphen) und im Knoten e angekommen ist. Ein Kriterium des Kriterienvektors könnte beschreiben, daß zum Finden einer schon bekannten Vorgeschichte nur die letzten drei Knoten seines Weges

berücksichtigt werden sollen, in diesem Fall c, d und e. Dieses Kriterium würde nun die Vorgeschichte zu *cde generalisieren, wobei * auf jeden Weg matcht, der im Startknoten beginnt und zum Knoten c führt. Ein ausführliches Beispiel ist in [Jordan 94] beschrieben.

4.5 Die Heuristik

Im Falle, daß die Generalisierung der beobachteten Benutzergeschichte (gemäß dem Kriterienvektor) auf keinen der bekannten Wege matcht, müssen die Kriterien des Vektors abgeschwächt werden, um zu einer allgemeineren Beschreibung zu gelangen. Zu diesem Zweck existiert eine Familie von Funktionen, eine für jedes Kriterium des Vektors, die als Heuristik bezeichnet wird. Jede Funktion schwächt ihr Kriterium bezogen auf die Häufigkeit des erfolgreichen Matchens und bezogen auf einen Faktor, der die Schrittweite der Abschwächung bestimmt. Dieser Faktor wird bei der Kalibrierung der Heuristik angepaßt. Die Konfliktlösungsstrategie wird benötigt, falls mehr als eine Vorgeschichte auf die beobachtete Vorgeschichte des Benutzers matcht. In diesem Fall muß eine einzige ausgewählt werden, die als Bedingung für die Auswahl einer bedingten Wahrscheinlichkeit dienen soll.

4.6 Adaption des Kriterienvektors und der Heuristik

Der Vorhersagemechanismus wird nach jedem Schritt des Schülers adaptiert. Dazu wird in jedem Schritt eine Prognose erstellt - auch wenn keine Hilfe angefordert wurde - und dann mit dem tatsächlichen Verhalten verglichen. Bei festgestellten Abweichungen werden Kriterienvektor und Heuristik angepaßt.
Als neuer Kriterienvektor wird ein Vektor eingesetzt, der die bisherige Vorgeschichte des Anwenders zu einer Vorgeschichte generalisieren würde, die, als Vorbedingung einer bedingten Wahrscheinlichkeit im Vorhersageknoten benutzt, die korrekte Vorhersage liefern würde. Die Anpassung der Heuristik basiert auf der Differenz zwischen altem und adaptiertem Kriterienvektor. Der Faktor für Schrittweite, der in jede Komponentenfunktion der Heuristik eingeht, wird so verändert, daß die Heuristik den zuvor gefundenen optimalen Kriterienvektor in einem Schritt liefern würde.

5. Hilfegenerierung

Studenten fordern dann Hilfen vom System an, wenn sie sich in einer Stocksituation befinden [Möbus et. al. 92a]. Sie wissen nicht, welchen Lösungsoperator sie einsetzen sollen. Hilfen sollten auf die aktuelle Situation abgestimmt sein und gerade so wenig Information enthalten, daß der Problemlöser die Stocksituation selbst überwinden kann. Desweiteren sollte die angebotene Information nur einen geringen Überraschungswert besitzen und damit kein neues Problem darstellen. Empirische Untersuchungen mit ca. 40 Studierenden haben gezeigt, daß beim Arbeiten mit Petri-Help Stocksituationen in der Anfangsphase der Problembearbeitung höchst selten auftreten. Hilfeinformation wird also erst dann benötigt, wenn das System sich schon an den jeweiligen Anwender anpassen konnte. Wie Erfahrungen zeigten, wurden Ergänzungsvorschlägen als hilfreich empfunden (siehe Abschnitt Petri Help). Eine gesteigerte Akzeptanz erwarten wir von benutzerangepassten Vorschlägen. Sie erfüllen weitgehend die oben aufgeführten Kriterien für sinnvolle Hilfen.

6. Generierung von Erklärungen

Wissenschaftliche Erklärungen geben u.a. Antwort auf Warum-Fragen [Groeben, Westmeyer 75]. Der zu erklärende Sachverhalt soll auf als bekannt vorausgesetzte Tatsachen und Gesetze zurückgeführt werden [Wahlster 81]. Bei den zugrundeliegenden Gesetzen kann man zwischen empirisch ermittelten und theoretisch fundierten Gesetzen unterscheiden (Bild 4). Die in Erklärungskomponenten von Expertensystemen [Puppe 88], natürlichsprachlichen Systemen [Wahlster 81] und Intelligenten Tutoriellen Systemen [Wenger 87, Herzog 93] eingesetzten Varianten basieren in der Regel auf einer Darstellung oder Aufbereitung der Ableitungsschritte des Systems [Puppe 88]. Wie oben ausgeführt läßt sich eine derartige „klassische" Komponente für Petri-Help nicht realisieren.

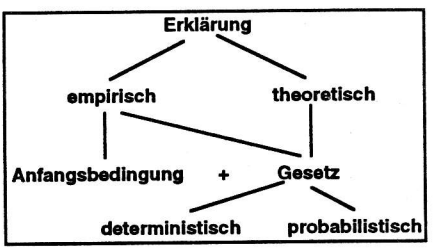

Bild 4: Mögliche Arten der Erklärung [Groeben, Westmeyer 75]

6.1 Gegenstand von Erklärungen in Petri-Help

Petri-Help bietet dem Benutzer unterschiedliche Formen von Informationen und Hilfen, die der Erklärung bedürfen:

- Liefert der Model Checker nach erfolgtem Hypothesentesten eine Fehlermeldung, d.h. ist eine (oder mehrere) der gewählten temporallogischen Formeln auf dem momentanen Netzentwurf nicht erfüllt benötigt der Benutzer Informationen darüber, warum seine Hypothese nicht erfüllt war.
- Fordert der Benutzer nach erfolglosem Hypothesentesten (s.o.) einen Modifikationsvorschlag seines Netzes an, so wird dieser Vorschlag darauf abzielen, die bislang nicht erfüllte Formel (-menge) zu erfüllen. Die Hinweise des Systems können dabei unterschiedlich umfangreich sein, u.U. wird nur ein erster Editierschritt auf dem Weg zu einem korrekten

Netz offeriert. Hilfen dieser Form müssen dem Anwender erklärt werden.

- Fragt ein Benutzer um Hilfe, ohne vorab eine fehlerhafte Hypothese getestet zu haben, kann ihm das System auch auf Zielebene einen Vorschlag zum weiteren Vorgehen unterbreiten. Dieser Vorschlag besteht in der Auswahl eines plausiblen nächsten Subzieles. Dieser Vorschlag des Systems muß dem Benutzer ebenfalls erklärt werden.

6.2 Basis für Erklärungen

Wegen des Fehlens von generativem Domänenwissen in Petri-Help müssen Erklärungen auf andere Informationsquellen zurückgreifen. Eine Möglichkeit der Erklärungsgenerierung basiert auf der Analyse des Verifikationsprozesses bzw. der dem Beweis zugrundeliegenden Daten (dem Fallgraphen). Auf diese Weise lassen sich Erklärungen der ersten und zweiten oben geschilderten Art ableiten.
Eine zweite Basis für Erklärungen sind empirische ermittelte Designheusitiken (Bild 5), die durch Auswertung von Videoprotokollen von Problemlösesitzungen sowohl mit Experten als auch mit Novizen entstanden [Möbus et. al. 93b].

6.2.1 Model Checking basierte Erklärungen

Basis für das Model-Checking ist der Fallgraph. Er stellt die möglichen Belegungen und Zustandsübergänge im Petri-Netz in Form eines endlichen Automaten dar. Bei dem Beweisprozeß wird die Formel rekursiv zerlegt und in den Knoten bzw. den Pfaden des Fallgraphen interpretiert. Schlägt nun ein Beweis fehl, können sowohl die Teile der Formel, als auch die Knoten oder Wege im Fallgraphen, die für das Scheitern verantwortlich sind, identifiziert werden. Durch den zergliedernden Charakter des Beweises ist es möglich, Fehlerbegründungen auf verschiedenen Ebenen anzugeben. Ist ein temporales Prädikat nicht realisiert worden, kann durch Animation im Netz die Schaltfolge angezeigt werden, die von der momentanen Belegung zu der Fehlersituation führt. Das ist in Petri-Help zwangsläufig der Fall, weil alle Formeln der Spezifikation durch Verwendung eines „Always"-Operators für alle Netzzustände erzwungen werden.
Eine Komponente, die diese Formen der Erklärung erzeugt, wurde bereits implementiert, eine Einbindung in das System Petri-Help und eine Akzeptanzprüfung der so generierten Erklärungen durch empirische Untersuchungen steht jedoch noch aus.
Erklärungen für Ergänzungs- und Korrekturvorschläge können ebenfalls aus dem Fallgraphen abgeleitet werden. Dazu werden die Graphen des fehlerhaften (bezogen auf eine Formel) und des korrigierten Netzes verglichen und die Unterschiede identifiziert. Diese Art der Erklärungsgenerierung wird zur Zeit im Rahmen einer Studienarbeit realisiert.

6.2.2 Erklärungen basierend auf Designheuristiken

Empirisch ermittelte Designheuristiken beschreiben den Zusammenhang zwischen einzelnen Formeln der Spezifikation und ihrer Realisierung durch ein Netzfragment oder die Reihenfolge der Umsetzung der Teile der Spezifikation durch Benutzer bezogen auf Eigenschaften der Formel und des Netzkontextes. Mit Hilfe dieser Heuristiken können Erklärungen für alle drei unter 6.1 aufgeführten Punkte erzeugt werden.
Liefert der Model-Checker beim Testen einer Hypothese einen Fehler, so kann versucht werden, ausgehend vom letzten als korrekt erkannten Netz, die Schritte zur versuchten Umsetzung der getesteten Formel(n) mit Hilfe der Heuristiken nachzuvollziehen. Stellt sich dabei heraus, daß zwar die Heuristiken korrekt verwendet wurden, aber der Netzkontext außer acht gelassen wurde, so kann die Angabe der verletzten Kontextbedingungen als Erklärung für das Scheitern des Beweises dienen.
Werden Ergänzungsvorschläge vom System als Hilfen angefordert, können die notwendigen Schritte ebenfalls durch Designheuristiken nachvollzogen werden. Als Erklärung kann dann entweder auf frühere Situationen verwiesen werden, in denen der Benutzer ebendiese Heuristiken erfolgreich einsetzte, oder es wird die ermittelte Heuristik angezeigt und der angebotene Ergänzungsvorschlag als ein erster Schritt hin zum Ergebnis der Anwendung dieser Heuristik erklärt.
Auf Zielebene schließlich kann durch Angabe einer Formelauswahlheuristik der Vorschlag eines nächsten Subzieles begründet werden.
Eine Komponente zur Generierung heuristik-basierter Erklärungen wird im Rahmen einer Studienarbeit entwickelt.

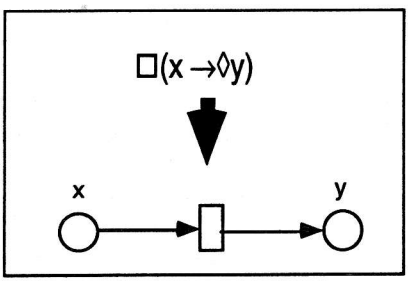

Bild 5: Designheuristik, die den Zusammenhang zwischen einer Formel und einem sie erfüllenden Netzfragment beschreibt

7. Validierung von Designheuristiken

Die Beobachtung von Petrinetzmodellierern legt den Schluß nahe, daß Menschen über verschiedene Designheuristiken verfügen, mit Hilfe derer sie temporallogische Formeln in Netze überführen [Möbus et. al. 93]. Ihre Vorgehensweise besteht häufig darin, einzelne Formeln der Spezifikation herauszugreifen und dann Netzfragment zu entwickeln, die genau jene Formel erfüllen. Die eingangs geschilderte Nicht-Monotonie wird dabei schlicht ignoriert. Natürlich kann nur durch nachträgliche Verifikation mittels Model-Checking festgestellt werden, ob der Lösungsentwurf tatsächlich der Spezifikation genügt. Um diese Designheuristiken zu validieren, müßte gezeigt werden, daß sie die Vorgehensweise einer größeren Anzahl Problemlöser adäquat beschreiben.

Ziel- und Aktionsgraph stellen die notwendigen Informationen bereit, um im Verhalten der Anwender nach Beziehungen zwischen Voraussetzung und Konklusio der Heuristiken zu suchen: Die Differenz zwischen zwei aufeinanderfolgenden Zielen im Zielgraphen entspricht der Vorbedingung von Heuristiken (temporallogische Formel), die Differenz zwischen den Netzen, die diese Ziele realisieren entspricht der Konklusio (Netzfragment). Der beschriebene Ansatz wurde bereits prototypisch implementiert und benutzt die Datenbasis, die bislang schon von Petri Help gelernt wurde. Dabei stellte sich heraus, daß mit den oben beschriebenen Designheuristiken die Problemlöseprozesse vieler Anwender beschreibbar sind.

8. Lernen von Designheuristiken

Es wurden Ansätze präsentiert, empirisch ermittelte Designheuristiken für die Generierung von Erklärungen einzusetzen. Ebenso wurde die Möglichkeit aufgezeigt, ihre Validität auf dem erhöbenen Benutzungsmodell zu prüfen. Es liegt nahe, das System in die Lage zu versetzen, selbständig Designheuristiken zu lernen. Dazu müßten „Heuristikskelette" vorgegeben werden, anhand derer das System nach plausiblen Korrespondenzen suchen könnte. Ergebnis wären dann valide Heuristiken, die einen (Groß-) Teil der beobachteten Benutzeraktionen beschrieben. Eine derartige Lernkomponente wird im Rahmen einer Diplomarbeit entwickelt.

9. Zusammenfassung

Es wurde gezeigt, wie ein wahrscheinlichkeitsbasierter Vorhersagemechanismus und eine adaptive Anpassung dieses Verfahrens zur Vorhersage von Benutzerverhalten in einem Intelligenten Tutoriellen System eingesetzt werden kann. Dabei wurde nicht auf eine Designtheorie der zu lehrenden Domäne zurückgegriffen, sondern der Suchraum wurde sukzessive aus bewerteten Benutzeraktionen gelernt. Zur Bewertung muß auf ein Orakel wie z.B. eine Verifikationskomponente zurückgegriffen werden. In diesem Zusammenhang wurde die enge Beziehung zwischen Planerkennung und Benutzermodellierung einerseits, und Planen und Verhaltensvorhersage andererseits aufgezeigt. Der Ansatz wurde ursprünglich zur Verbesserung von Petri Help entwickelt, er scheint jedoch auf andere Anwendungsbereiche übertragbar, die oben beschriebene Voraussetzungen erfüllen. Aus so gewonnenen Verhaltensvorhersagen können Hilfen erzeugt werden, die dann dem Benutzer erklärt werden müssen. Es wurden drei Klassen möglicher Erklärungen identifiziert und zwei Ansätze zur Generierung von Erklärungen vorgestellt. Petri-Help ist vollständig in MacProlog™ auf Macintosh™ Rechnern implementiert. Die verifikationsbasierte Generierung von Fehlererklärungen und die Validierungskomponente existieren als Prototyp. Der Benutzermodellierungs- und Vorhersagekomponente wird zur Zeit von Studenten eines Fortgeschrittenenpraktikums „Intelligente Tutorielle Systeme", die heuristikbasierte Erklärungsgenerierung im Rahmen zweier Studienarbeiten implementiert.

10. Literatur

[Ardissono et. al. 94] Ardissono, L., Lesmo, L., Sestero, D. (1994): Updating the User Model on the Basis of the Recognition of the User´s Plans, in: Proceedings of the Fourth International Conference on User Modeling (UM94), 15-19 August 1994, Hyannis, MA, USA, The MITRE Corporation, pp 5-10

[Bauer et. al. 91] Bauer, M., Biundo, S., Dengler, D., Hecking, M., Koehler, J., & Merziger, G., (1991): Integrated Plan Generation and Recognition - A Logic Based Approach, DFKI Research Report RR-91-26, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, Kaiserslautern/ Saarbrücken

[Buchanajew 94] Buchanajew, R. (1994): *Theorie der Stochastischen Automaten*, Teubner

[Claus 71] Claus, V. (1971): *Stochastische Automaten*, Teubner, Stuttgart

[Greer, McCalla 94] Greer, J., McCalla, G. (1994): Student Modelling: The Key to Individualized Knowledge-Based Instruction, Springer, Berlin

[Groeben, Westmeyer 75] Groeben, N. , Westmeyer, H. (1975): *Kriterien psychologischer Forschung*, Juventa Verlag, München

[Grunst et. al. 93] Grunst, G., Oppermann, R., Thomas, C. (1993): Benutzungsmodellierung bei kontext-sensitiver Hilfe und adaptiver Systemgestaltung, in: [Kobsa, Pohl 93]

[Herzog 93] Herzog, C. (1993): Generierung von Erklärungen verschiedener Ausprägungen und Detaillierungen in hierarchischen Wissensgebieten, in: C. Möbus: Workshop der Fachgruppe "Intelligente Tutorielle Systeme" der GI, 8./9. Juni 1993, Oldenburg , Interner Bericht des Fachbereichs Informatik, Universität Oldenburg

[Jordan 94] Jordan, O. (1994): Prognoseverfahren zur Verhaltensvorhersage, Studienarbeit, Fachbereich Informatik, Universität Oldenburg

[Josko 90] Josko, B. (1990). Verifying the correctness of AADL modules using model checking. In J.W. de Bakker, W.P. de Roever, G. Rozenberg (eds): Proceedings REX-Workshop on stepwise refinement of distributed systems: models, formalisms, correctness. Berlin: Springer, LNCS 430, 386-400.

[Kass 89] Kass, R. (1989): Student Modeling in Intelligent Tutoring Systems, in: [Kobsa, Wahlster 89]

[Kautz 91] Kautz, H. (1991): A Formal Theory of Plan Recognition and its Implementation, in: Allen, J., Kautz, H., Pelavin, R., & Tenenberg, J. (1991): *Reasoning about Plans.* San Mateo, Ca.: Morgan Kaufmann.

[Kobsa, Pohl 93] Kobsa, A. , Pohl, W. (eds.) (1993): Workshop 'Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen', KI93, Berlin, 13.-15.9.93. WIS- Memo 7 September 1993, AG Wissensbasierte Informations Systeme, Universität Konstanz, 1993

[Kobsa, Wahlster 89] Kobsa, A., Wahlster, W. (eds.) (1989):*User Models in Dialog Systems*, Berlin: Springer

[Koehn, Greer 94] Koehn, G., Greer, J., (1994): Recognizing Plans in Instructional Systems Using Granularity, in: Proceedings of the Fourth International Conference on User Modeling (UM94), 15-19 August 1994, Hyannis, MA, USA, The MITRE Corporation, pp 133-138

[Lusti 92] Lusti, M.: *Intelligente Tutorielle Systeme*, Oldenbourg Verlag, München, Wien, Handbuch der Informatik, Band 15.4

[McCalla, Greer 92] McCalla, G., Greer, J., (eds.) (1992): *Journal of Artificial Intelligence in Education*, Special Issue on Student Modelling, 3(4) 1992

[Möbus et al. 92a] Möbus, C., Pitschke, K., Schröder, O. (1992): Towards the theory-guided design of help systems for programming and modelling tasks. In C. Frasson, G. Gauthier & G.I. McCalla (eds): Intelligent tutoring systems, Proceedings ITS 92. Berlin: Springer, LNCS 608, 294-301.

[Möbus et al. 92b] Möbus, C., Schröder, O., Thole H.-J. (1992): A Model of the Acquisition and Improvement of Domain Knowledge for Functional Programming, in: [McCalla , Greer 92]

[Möbus et al. 93a] Möbus, C., Pitschke, K., Schröder, O., Folckers, J., Göhler, H. (1993): Erwerb von Modellierungswissen durch Hypothesentesten in PETRI-HELP, in: M. Sonnenschein, U. Lichtblau (Hrsg.), 5. Kolloquium der Arbeitsgruppe Informatiksysteme, Bericht Nr. AIS-10, FB Informatik der Universität Oldenburg, Mai 1993

[Möbus et al. 93b] Möbus, C., Pitschke, K., Schröder, O., Folckers, J., Göhler, H. (1993): PETRI-HELP - Intelligent Support for Petri Net Modellers, Interner Bericht, Fachbereich Informatik, Universität Oldenburg

[Olderog 91] Olderog, E.-R. (1991): *Nets, terms, and formulas*. Cambridge: Cambridge University Press

[Pitschke 93] Pitschke, K. (1993): Zielidentifikation und Planen in Intelligenten Tutoriellen Systemen. In: [Kobsa, Pohl 93]

[Pitschke 94] Pitschke, K. (1994): User Modeling for Domains without Explicit Design Theories, in: Proceedings of the Fourth International Conference on User Modeling (UM94), 15-19 August 1994, Hyannis, MA, USA, The MITRE Corporation, pp 191-195

[Puppe 88] Puppe, F. (1988):*Einführung in Expertensysteme*, Springer, Berlin

[Reisig 92] Reisig, W. (1992): *A primer in Petri net design*, Berlin: Springer

[Valiant 84] Valiant, L. (1984). A Theory of the Learnable. *Communications of the ACM*, 27, 1134-1142

[Wahlster 81] Wahlster, W. (1981): *Natürlichsprachliche Argumentation in Dialogsystemen*, Informatik-Fachberichte 48, Springer, Berlin, 1981

[Wedig 93] Wedig, A. (1993): Kalküle zur Entwicklung von Petri-Netzen aus Spezifikationen, Diplomarbeit am Fachbereich Informatik, Universität Oldenburg

[Wenger 87] Wenger, E. (1987): *Artificial Intelligence and Tutoring Systems*, Morgan Kaufmann Publishers, Los Altos, CA

[Wickens 82] Wickens, T. (1982): *Models for Behaviour - Stochastic Processes in Psychology*, W. H. Freeman and Company, San Francisco

AIS --- AIS --- AIS --- AIS

8. Kolloquium der Arbeitsgruppe Informatiksysteme

Michael Sonnenschein,
Hans Fleischhack (Hrsg.)

Bericht Nr. AIS-19 - Dezember 1994

Arbeitsgruppe Informatik-Systeme

FB Informatik - Universität Oldenburg

Herausgeber der Berichtsreihe:

Dr. Hans Fleischhack
Prof. Dr. Michael Sonnenschein

Anschrift der Autoren:

Fachbereich Informatik
Carl von Ossietzky Universität Oldenburg
Postfach 2503
26111 Oldenburg

Vorwort

Der vorliegende Bericht stellt die Ergebnisse dar, die beim siebten Kolloquium der Arbeitsgruppe Informatik-Systeme am 8. 7. 1994 präsentiert wurden.

Die Arbeitsgruppe Informatik-Systeme wurde mit dem Ziel gegründet, verschiedene Arbeitsbereiche des Fachbereichs Informatik stärker miteinander in Verbindung zu bringen. Dies soll unter anderem dazu dienen, methodische Grundlagenarbeit zur Vorbereitung des Oldenburger Forschungs- und Entwicklungsinstituts für Informatik-Werkzeuge und -Systeme (OFFIS) zu leisten.

Aus Mitteln der Stiftung Volkswagenwerk (Az. 210-70631/9-13-14/89) stehen der Arbeitsgruppe vom 1. 7. 1990 bis zum 31. 12. 1995 Mittel zur Finanzierung von 7,5 wissenschaftlichen Mitarbeiterstellen, Hilfskräften, Sachmitteln und Investitionen zur Verfügung.

Michael Sonnenschein
Hans Fleischhack