Erklärungsgenerierung in Petri-Help

Knut Pitschke, Olaf Schröder, Claus Möbus

Universität Oldenburg, Fachbereich Informatik

Abteilung Lehr-Lernsysteme, D-26111 Oldenburg

{pitschke, schroeder, moebus}@informatik.uni-oldenburg.de

Abstract

In dieser Arbeit werden, ausgehend vom wissenschaftlichen Erklärungsbegriff Formen der Erklärung und Erklärungsgenerierung in der Künstlichen Intelligenz eingeführt und diskutiert. Im Anschluß daran wird das Intelligente Hilfesystem Petri-Help vorgestellt und es werden Implikationen für die Erzeugung von Erklärungen abgeleitet. Schließlich werden Möglichkeiten der Erklärungsgenerierung unter den beschriebenen Prämissen aufgezeigt.

1. Wissenschaftliche Erklärungen

Der Begriff der Erklärung wird im Alltagssprachgebrauch in vielerlei Bedeutung verwendet. Wissenschaftliche Erklärungen werden jedoch verstanden "[..] als kausale Erklärungen von Vorgängen und Tatsachen. In diesem Fall besteht die Erklärung in der Angabe von Ursachen für bestimmte Sachverhalte" ([Groeben, Westmeyer 75], Kapitel 3), sie geben Antworten auf WARUM-Fragen. Erklärungen können klassifiziert werden nach dem Gegenstand der Erklärung (dem Explanandum) und der Art desjenigen, was erklärt (dem Explanans). Der Erklärungsgegenstand kann ein Ereignis oder eine Gesetzesaussage sein. Im ersten Fall spricht man auch von empirischen Erklärungen, im zweiten Fall von theoretischen Erklärungen. Empirische Erklärungen gehen von einer Anfangsbedingung und Gesetzesannahmen aus, theoretische Erklärungen stützen sich nur auf Gesetzesannahmen. Handelt es sich dabei um deterministische Gesetze, bezeichnet man die entsprechenden Erklärungen als *deduktiv-nomologisch*, liegen probabilistische Gesetze vor, so spricht man im Falle empirischer Erklärungen von *statistischen Analysen*, bei theoretischen Erklärungen von *deduktiv-statistischen* Erklärungen (siehe auch [Hempel 77]).

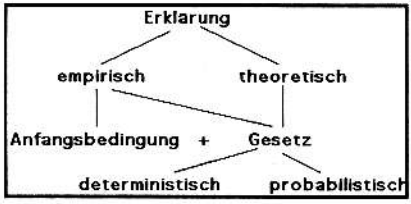

Abb. 1: Mögliche Erklärungen (nach [Groeben, Westmeyer 75])

2. Erklärungen in der KI

Erklärungen sind immer eingebettet in einen Erklärer-Versteher-Dialog [Schurz 92]. Ein solcher Dialog besteht mindestens aus einer nach Erklärung suchenden Frage und einer Antwort (der Erklärung), die die gestellte Frage beantwortet. Innerhalb der Künstlichen Intelligenz wird dabei (auch) untersucht, inwieweit die Rolle der Erklärers innerhalb eines solchen Dialoges von Computersystemen übernommen werden kann. Dies bietet sich an, weil in Systemen der Künstlichen Intelligenz Wissen über die zu bearbeitende Aufgabe bzw. die Domäne des Systems explizit repräsentiert ist. Dieses Wissen wird einerseits zur Ableitung von Lösungen eingesetzt, andererseits steht es aber auch für Erklärungszwecke zur Verfügung [Wick 94]. Eine große Klasse von Programmsystemen sind als Dialogsysteme konzipiert [Schmitt 83], d.h. der Benutzer eines solchen Systems löst seine Aufgabe in Interaktion mit dem Programm. Im Bereich

der KI fallen in diese Gruppe unter anderem Expertensysteme [Puppe 88], Planungs- und Konfigurierungssysteme [Hertzberg 89; Sauer 93] und Intelligente Tutorielle Systeme [Wenger 87; Lusti 92; Brna et al. 93]. [Puppe 88] bezeichnet die Erklärungsfähigkeit als eines der Hauptmerkmale, in denen sich Expertensysteme von konventionellen Computerprogrammen unterscheiden lassen. Dies liegt in der expliziten Trennung von Wissen und Problemlösestrategien begründet, die es ermöglicht, Problemlöseprozesse transparent darzustellen. So können Erklärungen in Expertensystemen Antworten auf WARUM-Fragen liefern, die ein Benutzer nach einer vom System vorgeschlagenen Diagnose oder Problemlösung stellt. Dabei wird, ausgehend von Anfangsbedingungen des Problemlöse-/Diagnoseprozesses die Folge der vom System unter Anwendung der Expertenregeln gezogenen Schlüsse dem Benutzer präsentiert.

Solche Erklärungen haben die Form deduktiv-nomologischer Ereigniserklärungen, falls es sich bei den im Expertensystem repräsentierten Regeln um deduktive Regeln handelt. Ist dagegen das Problemlösewissen in Form probabilistischer Gesetze repräsentiert, so handelt es sich bei den Erklärungen um statistische Analysen (nach [Groeben, Westmeyer 75]).

Nach einer Klassifikation in [Stoyan 92] können in Expertensystemen neben den klassischen WARUM-Erklärungen auch Erklärungen auf Fragen der Form WESHALB? (nach Gründen), WOZU? (nach dem Zweck), WIE? (nach Einzelheiten), ALTERNATIVEN?, WODURCH? (nach Rechtfertigungen), WAS BEDEUTET? und WAS IST GEMEINT? (nach Interpretationen) generiert werden. Die Antworten auf die meisten dieser Fragen basieren ebenfalls auf dem Ableitungsprozess des Expertensystems. Lediglich die WAS-Fragen zielen auf Faktenerklärungen ab.

[Wick, Thompson 92] kritisieren diesen Ansatz, weil sich die Generierung von Erklärungen auf die Darstellung des Schlußprozesses beschränkt [Graham, Jones 88]. Da der Ableitung der Lösung Regeln zugrunde liegen, die Expertenwissen repräsentieren und damit die Problemlösefertigkeiten von Experten widerspiegeln, können Erklärungen nur so "feinkörnig" sein, wie es die Expertenregeln sind (d.h. die Erklärungsschritte können nicht kleiner als die Ableitungsschritte des Expertensystems sein). Begründungen für einzelne Regeln können dementsprechend nicht geliefert werden. Es wird ein Ansatz vorgeschlagen, in dem mit Hilfe einer zweiten (Erklärungs-) Wissensbasis die vom Expertensystem erzeugte Lösung rekonstruiert und die Ableitungsschritte der Rekonstruktion als Erklärung verwendet werden. Anhand verschiedener Parameter kann vorab eingestellt werden, inwieweit die Erklärung zu der Expertensystemlösung korrespondieren soll. Dadurch ist es sowohl möglich, bei starker Kopplung die einzelnen Ableitungsschritte des Expertensystems durch feinere Zwischenschritte der Rekonstruktion zu erklären, als auch, bei loser Kopplung, alternative Ableitungswege darzustellen. Hierbei wird deutlich, daß die Erklärungsfähigkeit Wissen über die jeweilige Anwendungsdomäne (in diesem Fall diejenige des Expertensystems) benötigt. Unklar bleibt bei diesem Ansatz jedoch, warum das zusätzliche (Erklärungs- resp. Domänen-) Wissen nicht von vornherein dem Expertensystem zur Verfügung gestellt wird, um dadurch nicht nur die Erklärungsfähigkeit sondern auch die Problemlösefähigkeiten des Expertensystems zu verbessern.

Auch [Stoyan 92] sieht den Erklärungsprozeß in Expertensystemen als Deduktions- bzw. Beweisprozeß (ebendort S. 63). Dabei wird davon ausgegangen, daß derjenige, der um Erklärung ersucht, durch das vom Expertensystem erzielte Ergebnis überrascht ist. Durch schrittweises Nachvollziehen der Deduktionsschritte soll nun versucht werden, die Stelle zu finden, an der die Ableitungsprozesse von System und Anwender divergieren. Dies kann sowohl an fehlendem Fakten- oder Regelwissen seitens des Benutzers liegen, als auch an fehlerhaft angewendeten Regeln oder falschem Deduktionsziel. Ist die Stelle im Ableitungsprozeß erreicht, wo sich die Deduktionsketten von Benutzer und System unterscheiden, wird der Benutzer erneut überrascht. Entweder kann er den dargebotenen Schritt alleine nachvollziehen oder es entsteht erneut Erklärungsbedarf. Mit dieser Vorgehensweise kann allerdings immer nur der erste Fehler

(bzw. die erste Abweichung) in der Deduktionskette erklärt bzw. korrigiert werden und bei mehrfachen Fehlern, die sich gegenseitig aufheben, wird erst gar kein Erklärungsbedarf entstehen, da die Resultate ja übereinstimmen.

[Cawsey 92; 93] stellt ein System (EDGE, Explanatory Discourse Generator) zur interaktiven Erklärungsgenerierung vor. Als Domäne wurden dabei elektronische Schaltkreise gewählt. Dabei wird das Problem der Erzeugung von Erklärungen als ein Planungsproblem aufgefaßt. Dem System stehen Regeln zur inhaltlichen Planung von Erklärungen (content planning rules) und Regeln zur Gestaltung des Dialoges mit dem Benutzer (dialogue planning rules) zur Verfügung. Wird ein Erklärungsziel im Verlaufe der Interaktion mit dem Benutzer identifiziert, wird es in eine Planagenda eingetragen. Anschließend wird mithilfe der content planning rules versucht, Erklärungen für dieses Ziel zu finden. Dabei können u.U. Subziele identifiziert und entsprechend in die Agenda eingetragen werden. Mithilfe der dialogue planning rules wird dann für die gefundene Erklärung eine Präsentation für den Benutzer erzeugt. Die eigentliche Erklärungsgenerierung stützt sich dabei auf ein Benutzermodell, das als Overlay-Model bezogen auf das, in Form der content rules vorhandene Expertenwissen dargestellt wird. Die einzelnen Wissensitems werden als "bekannt", "unbekannt", "vielleicht bekannt" oder "unentscheidbar" klassifiziert. Diese Zuordnung erfolgt durch Inferenzen, die auf dem Dialogverhalten des Benutzers und auf den schon klassifizierten Wissensitems aufbauen. Die Planungsregeln entscheiden dann aufgrund der getroffenen Klassifikation, welche Wissensitems wie detailliert erklärt werden.

Ein Vorteil von Cawseys Ansatz ist seine Verständlichkeit und Einfachheit, "one major drawback of the EDGE system is the relative domain specifity of the content planning rules and the (related) limited treatment of discourse coherence" ([Cawsey 92], S. 87). Auch die Inferenzregeln, die die Inhalte des Benutzermodells bestimmen, scheinen stark domänenabhängig zu sein. So wird beispielsweise geschlossen: "IF all subconcepts are known THEN parent concept is known" ([Cawsey 92] S. 137). Es lassen sich leicht Beispiele konstruieren, bei denen ein Benutzer zwar alle Subkonzepte beherrscht, jedoch das Oberkonzept nicht kennt, weil ihm dieses Konzept irrelevant erscheint oder er nicht weiß, daß sich das Konzept in ihm bekannte Subkonzepte zerlegen läßt.

[Wahlster 81] beschreibt einen Ansatz zur natürlichsprachlichen Erklärungsgenerierung. Grundlage der erzeugbaren WARUM-Erklärungen ist die Rekonstruktion symbolischer Schluß- und Ersetzungsverfahren (ebendort S. 2/3). Zu diesem Zweck wird ein mehrsortiger Prädikatenkalkül entwickelt, der Mehrfachableitungen zu beschreiben gestattet. Den einzelnen Ableitungsschritten sind dabei Evidenzwerte zugeordnet, die zur Auswahl derjenigen Ableitungskette dienen, die als Basis der Erklärung dienen soll. Mit Hilfe eines ATN-basierten Formalismus (Augmented Transition Network) wird aus der gewählten Inferenz dann eine natürlich-sprachliche Satzkonstruktion erzeugt.

Die gemeinsamen Eigenschaften aller hier vorgestellten Ansätze zur Erklärungsgenerierung aus dem Bereich der KI lassen sich durch einen Satz von Wahlster zusammenfassen:

> "Erklärungskomponenten können lediglich Erklärungen für Ergebnisse von symbolischen Schluß- und Ersetzungsverfahren erzeugen, wie sie in pattern-gesteuerten Inferenzsystemen realisiert sind" ([Wahlster 81], S. 2).

Dies läßt sich begründen mit der offensichtlichen Analogie zwischen Argumentationsketten bei Erklärungen und Ableitungs- bzw. Inferenzketten bei regelbasierten Systemen. Ähnlich argumentieren Schank und Leake:

> "What constitutes a relevant explanation is well understood: it is simply a causal chain leading to the event ([Schank, Leake 90], S. 366).

Nach diesen Gesichtspunkten werden die später vorgestellten Ansätze zur Erklärungsgenerierung in Petri-Help bewertet werden.

3. Petri-Help

Petri-Help ist ein Hilfesystem, das Benutzer beim Erlernen von und bei der Modellierung mit Bedingungs-Ereignis Petri-Netzen unterstützt (Bild 3). Dem Lerner wird ein Folge von Modellierungsaufgaben 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.

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 2: Temporallogische Spezifikation der "Restaurant"-Aufgabe

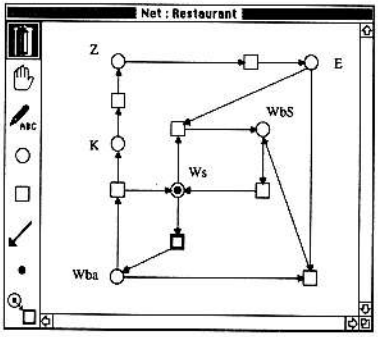

Bild 3: Lösung der "Restaurant"-Aufgabe

In Petri-Help werden die zu lösenden Aufgaben durch eine Menge temporallogischer Formeln spezifiziert, welche Eigenschaften der Lösung beschreiben (Bild 2). 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, nicht-monoton in Bezug auf die in ihnen erfüllten Formeln. 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. 93], 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. Erklärungen in Petri-Help

Studenten fordern dann Hilfen vom System an, wenn sie sich in einer Stocksituation befinden [Möbus et al. 92]. 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. In Petri-Help haben Hilfen die Form von Modifikationsvorschlägen (Ergänzungs- oder Korrekturvorschlägen) seitens des Systems, sie werden sowohl auf Planungsebene (Auswahl eines nächsten zu realisierenden Teilzieles) als auch auf Realisierungsebene (Petri-Netz Ebene) angeboten. Die Betonung liegt hierbei auf "angeboten", denn das System unterbricht den Lernenden nicht. Hilfestellung wird nur auf ausdrückliche Anforderung des Benutzers bereitgestellt. Die Hilfen basieren auf einer Vorhersage des Benutzerverhaltens die sich auf ein Benutzungsmodell und individuelle Kriterien des jeweiligen Benutzers stützt [Pitschke 94]. Die in den Hilfen dargebotene Information bedarf in der Regel der Erklärung. So ist es einem Lernenden nicht unbedingt einsichtig, warum er seinen Netzentwurf gerade um diejenige Stelle ergänzen soll, die ihm das System vorschlägt oder warum er ein bestimmtes Ziel (vorgegeben durch eine temporallogische Formel der Spezifikation) verfolgen soll. Desweiteren kann bei Fehlermeldungen nach Hypothesentesten (Auswahl und Überprüfung von Spezifikationsformeln auf dem Netzentwurf) Erklärungsbedarf entstehen, hierbei ist eine Begründung gefordert, warum eine als fehlerhaft klassifizierte Formel auf dem aktuellen Netzentwurf nicht erfüllt ist.

Erklärungsbedarf besteht also bei:

- Fehlerrückmeldungen des Model-Checkers nach Hypothesentesten
- angeforderten Ergänzungsvorschlägen
- Ergänzungsvorschlägen nach Fehlermeldungen

Im ersten der drei Fälle hat der Benutzer eine Hypothese bestehend aus einer oder mehreren Formeln über seinen Netzentwurf formuliert und vom System prüfen lassen. Die Rückmeldung vom System besagte nun, daß mindestens eine Formel nicht erfüllt sei. Eine Erklärung dieses Sachverhaltes müßte nun unterscheiden zwischen der Tatsache, daß er in seinem Netzentwurf einen Fehler hat, der verhindert, daß diese Formel erfüllt wird und dem Fall, daß sein Netz keinen unmittelbaren Fehler beinhaltet, sondern nur gewisser Ergänzungen bedarf, damit auch diese Formel erfüllt wird. Auf formaler Ebene ist es aber nicht entscheidbar, welcher der beiden

Fälle vorliegt. Das liegt auch in der Art der Verifikation begründet. Es kann immer nur überprüft werden, ob das entwickelte Netz das durch die temporallogischen Formeln spezifizierte Verhalten zuläßt, das heißt ein Modell der Formeln ist. Da es aber zu jeder Formel (oder jeder Menge von Formeln) abzählbar unendlich viele Modelle gibt, läßt sich eben nicht entscheiden, ob das vorliegende Netz Bestandteil eines dieser Modelle sein kann. Es kann jedoch auf Basis der vom System im Benutzungsmodell abgelegten Verhaltensweisen früherer Benutzer angegeben werden, ob ein Fall existiert, in dem ein Benutzer ein topologisch äquivalentes Netz erzeugt hat. Existiert ein solches Netz, dann können die beobachteten Wege zu einer Lösung dahingehend beurteilt werden, ob zum Erreichen einer (beobachteten) korrekten Lösung Teile des bislang erzeugten Netzes gelöscht werden mußten. Ist dies nicht der Fall, steht fest, daß es eine Fortsetzung ohne Löschhandlungen gibt, das vorhandene Netz also in eine korrekte Lösung einbettbar ist. In jedem Fall aber kann basierend auf einer Analyse des Verifikationsprozesses eine Erklärung generiert werden, warum die gewählte Formel auf dem erzeugten Netz nicht erfüllt ist.

Fordert der Benutzer vom System Ergänzungsvorschläge an, ist zu unterscheiden, ob zuvor eine Hypothese über gewählte Formeln getestet und als fehlerhaft zurückgewiesen wurde (in diesem Fall hätte der Vorschlag des Systems u.U. die Form eines Modifikationsvorschlages, siehe voriger Abschnitt) oder ob die Ergänzung vom Benutzer ohne vorherige, vom System erkennbare Stocksituation angefordert wurde.

Im Fall, daß der Ergänzungsanforderung keine fehlerhafte Hypothese vorausging, muß eine zielgerichtete Erklärung bereitgestellt werden, d.h. dem Lernenden muß klar gemacht werden, warum gerade dieser Vorschlag des Systems ihn näher an sein gewähltes (Sub-)Ziel bringt. In diesem Fall ist ein Erklärung basierend auf Heuristiken denkbar.

Ging dem Hilfeersuchen des Benutzers eine fehlerhafte Hypothese voraus, so ist der Vorschlag des Systems darauf ausgelegt, diesen Fehler zu beheben oder, falls die Fehlerbeseitigung eine Reihe von Handlungen umfaßt, einen ersten Schritt auf diesem Wege anzubieten. Zur Erklärung muß dann die gleiche Entscheidung getroffen werden wie im Falle der Fehlererklärung. Lag ein Fehler im Netz vor (d.h. das System kennt keine Fortsetzung zu einer korrekten Lösung, die ohne Löschhandlungen auskommt), kann aufgrund einer Analyse der Fallgraphen (des momentanen Netzes und des modifizierten Netzes, das u.U. auch mehrere Schritte entfernt sein kann) erklärt werden, wie sich die vorgeschlagene Modifikation auf die Dynamik des Netzes auswirkt.

Wenn im Netzentwurf kein Fehler gemacht wurde, sondern er nur unvollständig bezogen auf die getestete Hypothese war, kann mit Hilfe von Heuristiken eine Erklärung erzeugt werden.

4.1 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. Dazu wird zuerst der Fall im Netz angegeben, in dem der „Always"-Operator, der in Petri-Help jede Formel einschließt, verletzt wurde. Auf Wunsch kann die Schaltfolge des Netzes, ausgehend von der Anfangsbelegung, animiert werden, die zu der Situation führte, in welcher der durch den „Always"-Operator gebundene Teil der Formel nicht erfüllt ist. Genügt dem Lernenden diese Erklärung nicht, so wird die Formel weiter zerlegt und es wird angegeben, welche Teile in der oben beschriebenen Situation nicht erfüllt sind. Werden bei dieser Zerlegung weitere durch temporale Prädikate gebundene Teile extrahiert, so ist wiederum eine Animation der entsprechenden Schaltfolge möglich.

Eine Komponente, die diese Formen der Erklärung erzeugt, wurde bereits implementiert und in das System Petri-Help eingebunden, eine Akzeptanzprüfung der so generierten Erklärungen durch empirische Untersuchungen steht jedoch noch aus.

Bild 4 (a-e) zeigt ein Beispiel für eine Sequenz unterschiedlich abstrakter Erklärungen, die auf diese Weise erzeugt wurden. Zu Anfang (Bild 4a) wird nur die Information gegeben, daß eine Always-quantifizierte Formeln in jedem Netzzustand gelten muß, dies aber nicht der Fall ist. Klickt der Benutzer "WEITER", wird erklärt, daß es einen Zustand des Netzes gibt, in dem zwar Prämisse, nicht jedoch Konklusio der Implikation erfüllt ist (Bild 4b). In Bild 4c wird der Grund für das Scheitern der Konklusio angegeben. Eine Netzsimulation, die auf Anforderung parallel dazu im Petri-Netz-Editor vorgenommen wird, verdeutlicht die Schaltfolge des Netzes, die zu der letztendlichen Fehlersituation führte (Bilder 4d, 4e).

Bild 4a

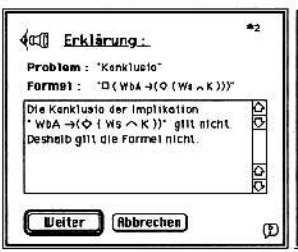

Bild 4b

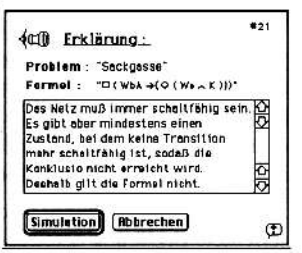

Bild 4c

Bild 4d

Bild 4e

Bild 4 (a-e): Verschiedene Stadien einer verifikationsbasierten Erklärung

Nur indirekt auf dem Model-Checking basieren Erklärungen für Ergänzungs- und Korrekturvorschläge die aus dem Fallgraphen abgeleitet werden können. Dazu werden die Graphen des fehlerhaften (bezogen auf eine Formel) und des korrigierten Netzes verglichen und die Unterschiede identifiziert. Mit Hilfe der Fallgraphen kann wiederum eine Simulation angeboten werden, die die Unterschiede in den beiden Netzen verdeutlicht und aufzeigt, warum eine vorher unerfüllte Formel nach dem Einfügen der vorgeschlagenen Ergänzung in dem Netz gilt. Diese Art der Erklärungsgenerierung wird zur Zeit im Rahmen einer Studienarbeit realisiert.

Model-Checking-basierte Erklärungsgenerierung fällt also ebenso in die Klasse derjenigen in Abschnitt 2 vorgestellten Ansätze. Ein Unterschied besteht lediglich darin, daß nicht die Ableitungsschritte, die zur Herleitung einer Lösung eingesetzt wurden, zu Erklärungszwecken benutzt werden (wie in Expertensystemen), sondern Ableitungsschritte eines formalen Beweises zur Veranschaulichung derjenigen Eigenschaften der Lösung dienen, die entweder zu einem Fehler führten oder die durch eine vorgeschlagene Ergänzung zusätzlich erfüllt werden.

4.2 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. Um mit Hilfe dieser Heuristiken sinnvolle Erklärungen für Ergänzungs- und Modifikationsvorschläge erzeugen zu können, müssen diese Heuristiken Konzeptwissen unterschiedlichen Abstraktionsniveaus beinhalten.

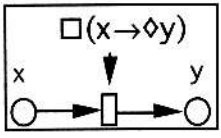

Bild 5: Designheuristik, die den Zusammenhang zwischen einer Formel und einem sie erfüllenden Netzfragment beschreibt

Heuristikbasierte Erklärungen können sowohl dann sinnvoll eingesetzt werden, wenn im vorhandenen Netz, für das ein Ergänzungsvorschlag angefordert wurde, kein Fehler vorlag (siehe Kap. 4), d.h. das System kennt eine Fortführung des Netzes zu einer korrekten Lösung ohne Löschhandlungen, als auch in Situationen, wo ein Fehler im Netz identifiziert wurde. Im letzteren Fall können Reparaturheuristiken zur Erklärungsgenerierung benutzt werden.

Erklärungen sollen begründen, warum der Vorschlag des Systems im momentanen Kontext sinnvoll ist. Dazu reicht es nicht, nur einen (z.B. heuristikbasierten) Ableitungsschritt anzugeben, der zu der vorgeschlagenen Situation führt. Stattdessen sollten abstraktere (und u.U. auch generellere) Konzepte präsentiert werden.

Es sind drei heuristikbasierte Erklärungsformen vorgesehen:

- Darstellen derjenigen Konzepte, zu deren Realisierung der Vorschlag einen ersten Schritt darstellt
- Aufzeigen von analogen Situationen, in denen der Lernende ebenso verfahren ist
- Herausstellen von verletzten bzw. unerfüllten Konzepten

In den beiden ersten Fällen werden Heuristiken verwendet, um den dargebotenen Ergänzungsschritt nachzuvollziehen. Gelingt dies, werden Heuristiken extrahiert, die diesen Schritt mittels abstrakter Konzepte beschreiben. Diese abstrakteren Konzepte als Erklärung präsentiert, erleichtern es dem Benutzer, Ähnlichkeiten zwischen seinem konkreten und vergleichbaren, vielleicht schon gelösten Problemen zu sehen.

Im dritten Fall können, ausgehend von Fehlermeldungen des Model-Checkers und dem Ergänzungsvorschlag des Systems mit Hilfe von Reparaturheuristiken verletzte Konzepte identifiziert werden.

Auf Zielebene schließlich kann durch Angabe einer Formelauswahlheuristik der Vorschlag eines nächsten Subzieles begründet werden.

Eine Komponente zur Generierung heuristik-basierter Erklärungen wird zur Zeit im Rahmen einer Studienarbeit entwickelt.

5. Lernen von Designheuristiken

Wie oben beschrieben, sollen Designheuristiken das Problemlöseverhalten von Benutzern widerspiegeln. Um sie zur Erklärungsgenerierung einsetzen zu können, ist es unabdingbar, daß sie neben konkreten Aktionen zur Umsetzung eines Zieles auch abstraktere Beschreibung von Konzepten enthalten.

Das Benutzungsmodell wird aufgebaut aus dem beobachteten Verhalten aller Benutzer, die mit dem System arbeiten. Daraus werden bedingte Wahrscheinlichkeiten abgeleitet, die Übergänge

von beobachtbaren Zuständen während der Aufgabenbearbeitung ineinander bezogen auf vorherige Verhaltensweisen beschreiben. Diese Übergangswahrscheinlichkeiten, zusammen mit einem adaptiven Ähnlichkeitsmaß, das die Übereinstimmung des Verhaltens des konkreten Benutzers mit den dem System bekannten Verhaltensweisen widerspiegelt, bilden die Basis für Modifikationsvorschläge seitens des Systems.

Darüberhinaus kann das Benutzungsmodell auch als Grundlage zur Extraktion von Designheuristiken verwendet werden. Ein erster wissensarmer Ansatz ging davon aus, daß Korrelationen zwischen Zustandsübergängen im Problem- bzw. Lösungsraum (innerhalb des Benutzungsmodells aufgebaut) Indikatoren für mögliche Heuristiken liefern können. Damit lassen sich zwar plausible (im Sinne Lenats und Clanceys, [Lenat 83; Clancey 85]) Problemlöseoperatoren finden, um diese aber zur Erklärungsgenerierung einsetzen zu können, ist hierarchisch strukturiertes, domänenspezifisches Konzeptwissen nötig. Diese Form von Wissen ist notwendig, um in Erklärungen Bezug nehmen zu können auf domänenspezifische Begriffe.

Zur Zeit wird daher ein Ansatz verfolgt, der, ausgehend von Konzepthierarchien und vorgegeben Indikatoren für Plausibilität (z.B. Häufigkeit des erfolgreichen Einsatzes) versucht, mit Hilfe eines Lernverfahrens aus dem Bereich des Conceptual Clustering [Fisher, Langley 86], Gruppen von Zustandsübergängen (im Problemraum) zusammenzufassen und einheitlich in Form eines Konzeptbegriffes zu beschreiben [Jordan 95].

6. Zusammenfassung

In dieser Arbeit wurden Ansätze der Erklärungsgenerierung aus dem Bereich der KI vorgestellt und verglichen. Es wurde aufgezeigt, wie in einer intelligenten Problemlöseumgebung - Petri-Help - ohne explizit repräsentiertes Designwissen Erklärungen für erzeugte Hilfen oder Fehlermeldungen generiert werden können. So können sich Erklärungen auf eine Analyse des formalen Verifikationsprozesses stützen oder sie können unter Verwendung von Designheuristiken erzeugt werden. Diese Heuristiken können vom System auf Basis vorgegebener Konzepthierarchien gelernt werden. Eine Komponente, die Erklärungen basierend auf einer Analyse der Verifikation erzeugt wurde bereits implementiert und in Petri-Help eingebunden. Erste (nicht repräsentative) Versuche mit Studierenden im Hauptstudium Informatik haben eine gute Akzeptanz ergeben. Eine Evaluation steht jedoch noch aus. Komponenten zur heuristikbasierten Erklärungsgenerierung und zur Akquisition von Heuristiken befinden sich zur Zeit im Stadium der Implementation.

7. Anmerkung

Petri-Help wird gefördert durch die Stiftung Volkswagenwerk (Az. 210-70631/9-13-14/89). Wir danken Jörg Folckers für die Implementierung großer Teile Petri-Helps.

8. Literatur

[Brna et al. 93] P. Brna, S. Ohlsson, H. Pain (eds.): Proceedings AI-ED 93, World Conference on Artificial Intelligence and Education, Edinburgh, AACE, 1993

[Cawsey 92] A. Cawsey: Explanation and Interaction, The Computer Generation of Explanatory Dialogues, MIT Press, Cambridge, MA, 1992

[Cawsey 93] A. Cawsey: User Modelling in Interactive Explanations, User Modeling and User-Adapted Interaction, Vol.3, No.3, pp 221-247, 1991

[Clancey 85] W. Clancey: Heuristic Classification, Artificial Intelligence, Vol.27, no.3, pp 289-350, 1985

[Fisher, Langley 86] D. Fisher, P. Langley: Conceptual Clustering and Its Relation to Numerical Taxonomy, in: W. Gale (ed.): Artificial Intelligence and Statistics, Addison-Wesley, Reading, MA, 1986

[Graham, Jones 88] I. Graham, P. Jones: Expert Systems: Knowledge, Uncertainty and Decisions, Chapman and Hall, London, New York, 1988

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

[Hempel 77] C. Hempel: Aspekte wissenschaftlicher Erklärungen, de Gruyter, Berlin, 1977

[Hertzberg 89] J. Hertzberg: Planen, BI-Wissenschaftsverlag, Mannheim, 1989

[Jordan 95] O. Jordan: Lernen von Designheuristiken in Petri-Help, Diplomarbeit, Fachbereich Informatik, Uni Oldenburg, in Arbeit, 1995

[Josko 90] B. Josko: 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, Springer, Berlin, LNCS 430, S. 386-400, 1990

[Lenat 83] D. Lenat: The Nature of Heuristics, Artificial Intelligence, Vol. 19, pp 189-249, 1983

[Lusti 92] M. Lusti: Intelligente Tutorielle Systeme, Oldenbourg Verlag, München, Wien, Handbuch der Informatik, Band 15.4, 1992

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

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

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

[Pitschke 94] K. Pitschke: 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, 1994

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

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

[Sauer 93] J. Sauer: Wissensbasiertes Lösen von Ablaufplanungsproblemen durch explizite Heuristiken, DISKI-Reihe, infix Verlag, St. Augustin, 1993

[Schank, Leake 90] R. Schank, D. Leake: Creativity and Learning in a Case-Based Explainer, in: J. Carbonell (ed.): Machine Learning, Paradigms and Methods, MIT Press, Cambridge, MA, 1990

[Schmitt 83] A. Schmitt: Dialogsysteme: kommunikative Schnittstellen, Softwareergonomie und Systemgestaltung, BI-Wissenschaftsverlag, Mannheim, 1983

[Schurz 92] G. Schurz: Erklärungsmodelle in der Wissenschaftstheorie und in der Künstlichen Intelligenz, in: H. Stoyan (Hrsg.): Erklärung im Gespräch - Erklärung im Mensch-Maschine-Dialog, Springer, Berlin, Informatik-Fachberichte 310, 1992

[Stoyan 92] H. Stoyan: Erklärungen und Beweise, in: H. Stoyan (Hrsg.): Erklärung im Gespräch - Erklärung im Mensch-Maschine-Dialog, Springer, Berlin, Informatik-Fachberichte 310, 1992

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

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

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

[Wick 94] M. Wick: Explanation as a Primary Task in Problem Solving, The Knowledge Engineering Review, Vol. 9: 1, pp 78-82, 1994

[Wick, Thompson 92] M. Wick, W. Thompson: Reconstructive expert system explanation, Artificial Intelligence 54, pp 33-70, 1992

Friedbert Huber-Wäschle Helmut Schauer
Peter Widmayer (Hrsg.)

GISI 95

Herausforderungen eines globalen Informationsverbundes für die Informatik

25. GI-Jahrestagung und
13. Schweizer Informatikertag
Zürich, 18.-20. September 1995

Springer

Herausgeber

Friedbert Huber-Wäschle
Peter Widmayer
Institut für Theoretische Informatik
ETH Zentrum
CH-8092 Zürich

Helmut Schauer
Institut für Informatik, Universität Zürich
Winterthurer Strasse 190, CH-8057 Zürich

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Herausforderungen eines globalen Informationsverbundes für die Informatik : 25. GI-Jahrestagung und 13. Schweizer Informatikertag, Zürich, 18. - 20. September 1995 / GISI '95. Friedbert Huber-Wäschle ... (Hrsg.). GI. - Berlin ; Heidelberg ; New York ; London ; Paris ; Tokyo ; Hong Kong ; Barcelona ; Budapest : Springer, 1995
(Informatik aktuell)
ISBN 3-540-60213-5
NE: Huber-Wäschle, Friedbert [Hrsg.]; GISI <1995, Zürich>; Schweizer Informatikertag <13, 1995, Zürich>

CR Subject Classification (1995): A.0

ISBN 3-540-60213-5 Springer-Verlag Berlin Heidelberg New York

Satz: Reproduktionsfertige Vorlage vom Autor/Herausgeber
Druck- u. Bindearbeiten: Weihert-Druck GmbH, Darmstadt
SPIN: 10484670 33/3140-543210 – Gedruckt auf säurefreiem Papier