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

3. Kolloquium der Arbeitsgruppe Informatik-Systeme

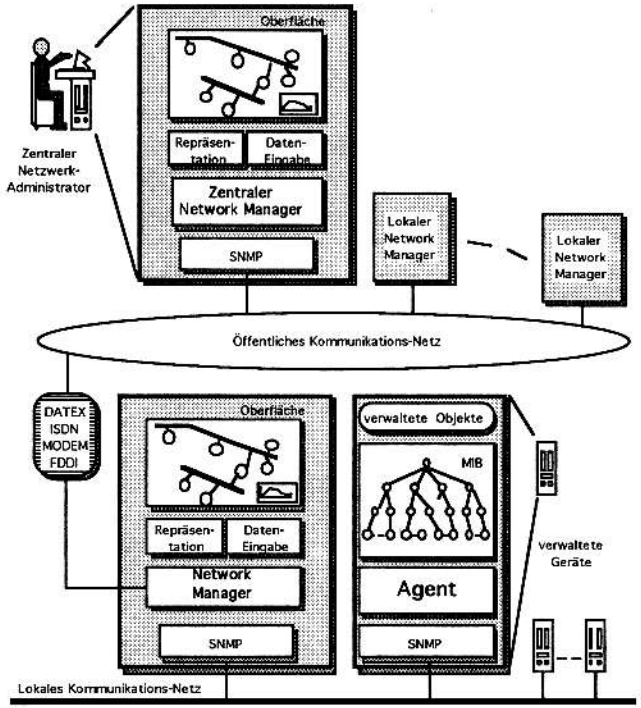
Volker Claus, Ulrike Lichtblau (Hrsg.)

Bericht Nr. AIS-5 - April 1992

Arbeitsgruppe Informatik-Systeme

FB Informatik - Universität Oldenburg

zu werden. Erst wenn der lokale Manager nicht mehr über ausreichende Mittel verfügt, ein Problem zu bewältigen, wird der zentrale Manager informiert. Dieser hat entweder genügend Information, das Problem automatisch zu lösen, oder er informiert einen zentralen Netzwerkadministrator, der manuell, gegebenenfalls auch vor Ort, das Problem bearbeitet.



Eine weitere Hierarchisierung kann außer in der angegebenen Anwendung auch dann nützlich sein, wenn ein einzelner Manager nicht mehr imstande ist, die gesamte anfallende Information zu verwalten. Durch Hierarchisierung kann das Management auch auf größere Systeme, bei denen das SNMP-Konzept erfahrungsgemäß versagt, ausgeweitet werden.

Gewinnung von Planregeln und Designheuristiken für PETRI-HELP

Claus Möbus, Knut Pitschke, Olaf Schröder, Hermann Göhler
Carl-von-Ossietzky-Universität Oldenburg, Fachbereich 10, Angewandte Informatik,
Abteilung Lehr- Lernsysteme

Einführung

Im Rahmen unseres Projektes PETRI-HELP wird ein intelligentes Hilfesystem entwickelt, das Benutzer bei der Modellierung von Petrinetzen auf verschiedenen Entwurfsebenen mit wissensstandsbezogenen Hilfen unterstützt. Dabei soll weder der Novize überfordert, noch der erfahrene Modellierer durch überflüssige oder zu detaillierte Informationen gestört oder behindert werden [Möbus, Pitschke, Schröder 91a, 91b].

Die Entwicklung wissensstandsbezogener Hilfen erfordert detaillierte Annahmen über das aktuelle Wissen des Benutzers. Daher ist es notwendig, das Hilfesystem auf einer kognitionspsychologischen Theorie des Wissenserwerbs und der Hilfen zu gründen. Hierzu entwickelten wir die ISPDL-Theorie (*I*mpasse - *S*uccess - *P*roblem solving - *D*riven *L*earning, [Möbus 91], [Möbus, Schröder, Thole 91], die Aspekte der Handlungsstrukturierung [Gollwitzer 90], lerntheoretische Aspekte des Impasse Driven Learning [vanLehn 88], [vanLehn 90], [vanLehn 91] und des Success Driven Learning [Anderson 89], [Wolff 87]) miteinander verbindet (Abb. 1, zur Erläuterung siehe [Möbus 91]). Danach ist der Problemlöser bevorzugt in *Stocksituationen* bereit, durch *Heuristiken* wie die Nutzung von Hilfen neues Wissen aufzunehmen: Ohne Stocksituationen kein Informationsbedarf. Hat der Problemlöser Erfolg, so wird er Konzepte, die zur Lösung führten, später wieder verwenden. Die ISPDL-Theorie führt zu folgenden Designprinzipien für das Hilfesystem [Möbus, Pitschke, Schröder 91c]:

- Das System sollte Hilfen *anbieten*, den Problemlöser jedoch nicht unterbrechen.
- Der Problemlöser muß *jederzeit* Gelegenheit zu detaillierter Hilfeinformation haben.
- Hilfen müssen auf verschiedenen *Ebenen* (Planung, Ausführung, Bewertung) erfolgen.
- Die angebotenen Hilfen müssen sich an dem *aktuellen Lösungsplan* des Problemlösers orientieren.
- Die angebotenen Hilfen müssen sich an dem *aktuellen Stand des Domänenwissens* des Problemlösers orientieren. Dazu ist ein Wissensstandsmodell ("*Stadienmodell*") erforderlich.
- Das Stadienmodell muß durch ein *Prozeßmodell* des Erwerbs und der Modifikation von Wissen, des Einsatzes von Heuristiken etc. gestützt werden.
- Der Lernende muß sich frei in dem System bewegen können, damit Daten zur Bildung von Hypothesen über kognitive Prozesse gewonnen werden können.

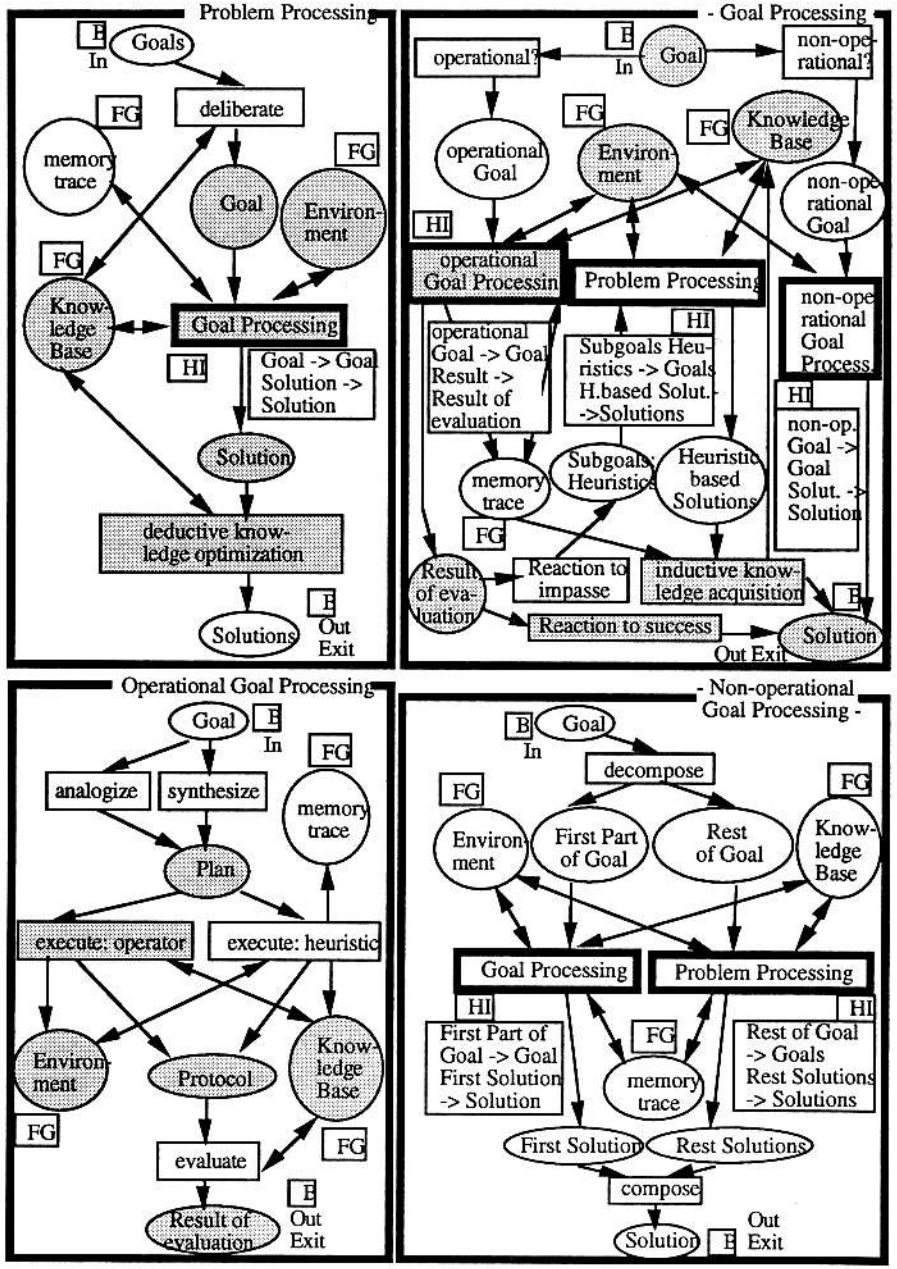


Abb. 1: Die ISPDL-Theorie, dargestellt als höheres hierarchisches Petrinetz

Im zweiten Halbjahr 1991 wurde an folgenden Aspekten gearbeitet:

- Die Entwicklung, geeignete Formulierung und Analyse einer Menge von Planungsaufgaben für die Entwicklung von Bedingungs-Ereignis-Netzen.
- Die Analyse von Benutzerlösungen.
- Die Entwicklung von Benutzerhilfen in Form von Planregeln und Designheuristiken.

Entwicklung und Analyse von Aufgabenstellungen

Die Analyse von Lösungsentwürfen und die Bereitstellung von Hilfen für Petrinetzmodellierer ist nur sinnvoll, wenn die jeweilige Aufgabenstellung klar spezifiziert ist. Wir bieten eine Aufgabenbeschreibung mit temporallogischen Formeln an, mit Hilfe derer das intendierte Verhalten eines Netzes (bzw. einer Menge von Netzen) beschrieben wird. Damit wird die Entwurfsanalyse durch das System ermöglicht. Diese Formelmenge ist Basis für eine Verifikationskomponente (in Anlehnung an [Josko 90]), welche die Netze auf Erfüllung dieser Formeln hin überprüft (s.u.). Für eine Sequenz von gegenwärtig 15 Übungsaufgaben wurden temporallogische Spezifikationen entwickelt. Beispiele für die temporallogisch spezifizierten Übungsaufgaben zur Modellierung mit Bedingungs-Ereignis-Netzen (BE-Netze) sind die Benutzung eines Telefons, der Ablauf der Photosynthese, das Verhalten eines Thermostaten usw.

Die 15 Übungsaufgaben lassen sich gemäß einer Sequenz von fünf Lernzielen partiell ordnen: Entwicklung von BE-Netzen für Implikationen mit

a) nur atomaren Formeln in Prämisse und Conclusio,
b) Konjunktionen,
c) Berücksichtigung von Kontext (gleiche Ausdrücke in Prämisse und Conclusio),
d) Disjunktionen, und
e) zusätzlicher Berücksichtigung von Ausschlußbedingungen.

Abb. 2 zeigt die partielle Ordnung der 15 Aufgaben in der Lernzielhierarchie. (Z.B. bezieht sich die Aufgabe "Thermostat" auf die Lernziele b und c.) Abb. 3 zeigt als Beispiel für eine Aufgabe eine temporallogische Spezifikation und einen Netzentwurf. Diese Aufgabe ("Telefon") umfaßt die Lernziele a, b und d.

In mehreren empirischen Einzelversuchen mit dem Petrinetz-Editor der MOBY-Projektgruppe haben wir den Personen temporallogische Formelmengen wie in Abb. 3 als Aufgabenstellungen präsentiert. Zusätzlich wurde den Benutzern eine Erklärung der temporallogischen Symbole ausgehändigt. Die Ergebnisse ließen umgangssprachliche Aufgabenformulierungen zusätzlich zu den temporallogischen Formulierungen nicht als notwendig erscheinen.

Analyse von Benutzerlösungen

Die vom Lernenden entwickelten Lösungsentwürfe müssen vom System online analysiert werden, um

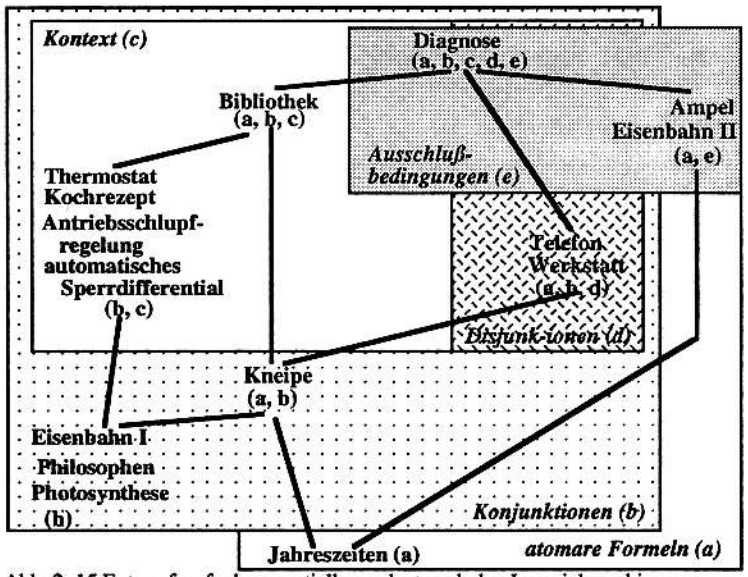


Abb. 2: 15 Entwurfsaufgaben, partiell geordnet nach den Lernzielen a bis e

- über ihre Korrektheit zu entscheiden,
- das aktuelle hypothetische Domänenwissen des Lernenden (= Wissen zur Überführung spezifizierter Aufgaben in BE-Netze) zu identifizieren (Wissensdiagnose) und
- auf der Basis des aktuellen Wissensstands wissensstandsbezogene Hilfen bereitzustellen.

Daher werden während des Editierprozesses alle erzeugten stellengeränderten Petrinetze mit Hilfe eines Model Checkers auf Erfüllung der Formeln der Aufgabenspezifikation hin überprüft. Es kann also festgestellt werden, welcher Menge temporallogischer Formeln das gerade existierende Netz genügt. Da aber die Erstellung von Petrinetzen in Bezug auf die von ihnen erfüllten Formeln nicht-monoton ist, wurde eine Klassifikation eingeführt, die es gestattet, sogenannte "sichere" Zustände (Zustand = Netz) zu identifizieren.

Bereitstellung von Hilfen

Auf der Basis sicherer Zustände können dem Benutzer verschiedene Formen von Hilfen angeboten werden.

- *Designregeln*, die das momentane Netz in das nächste sichere Netz auf dem Weg zu einer Lösung überführen. Diese Regeln wurden vom System in früheren Sitzungen gelernt.
- *Designregeln*, die ein sicheres Netz mit der Menge von Formeln assoziieren, die in ihm erfüllt werden. Auch diese Form von Regeln wird vom System automatisch generiert.

Ein Lösungsvorschlag besteht nun in einem Netz, dessen assoziierte Formelmenge eine (kleinste) Obermenge der zur Zeit erfüllten Formeln bildet.

Ferner können wir den Benutzern heuristische Designbausteine anbieten, die in empirischen Einzeluntersuchungen gewonnen wurden:

- *Designbausteine*, die Formeln in Netzteile überführen. Sie gestatten es dem Anwender, Formeln zu wählen, die er als nächstes erfüllen möchte. Wegen der Nichtmonotonie bei der Konkatenation von Netzen ist hier jedoch eine anschließende Korrektheitsprüfung durch den Model-Checker nötig.

- [] P1 will tel. ∧ Tel. ruhig -> ◊ (P1 hebt Hörer ab ∨ Tel. klingelt)
- [] P1 hebt Hörer ab -> ◊ P1 hört Dauerton
- [] P1 hört Dauerton -> ◊ P1 wählt
- [] P1 wählt -> ◊ (kein Anschluß ∨ P2 nimmt ab ∨ frei ∨ besetzt)
- [] P1 ignoriert Tel. ∨ Gespräch P1-P2 ∨ kein Anschluß ∨ frei ∨ besetzt -> ◊ Tel. ruhig
- [] Tel. klingelt -> ◊ (P1 nimmt ab ∨ P1 ignoriert Tel.)
- [] P1 nimmt ab ∨ P2 nimmt ab -> ◊ Gespräch P1-P2
- [] Tel. ruhig -> ◊ Tel. klingelt

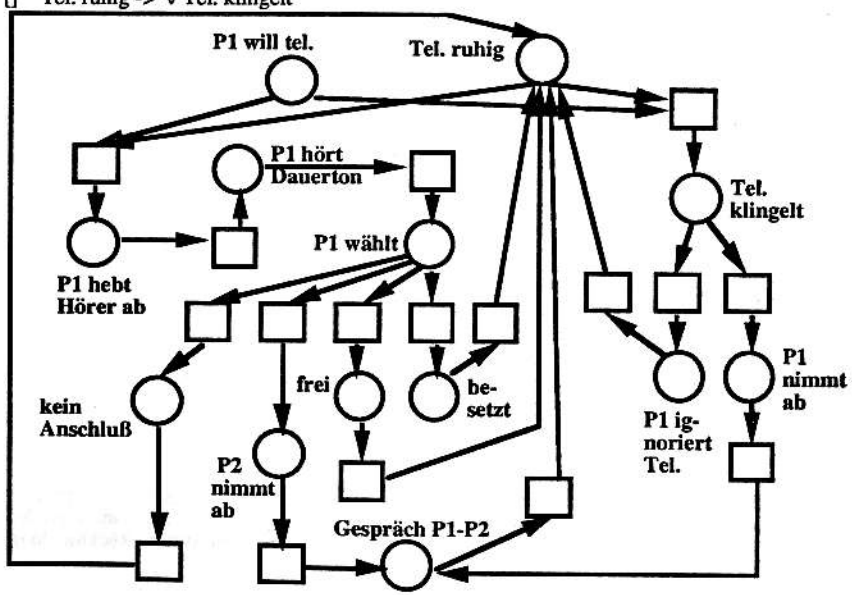


Abb. 3: Formale Beschreibung und empirischer Netzentwurf für die Aufgabe "Telefon" (in Anlehnung an [Müller 88])

Die Designbausteine wurden in empirischen Einzeluntersuchungen mit insgesamt 14 Personen gewonnen. Die Personen bearbeiteten in den Einzelsitzungen einen Teil der o.g. 15 Aufgaben am MOBY-Netzeditor. Jede Formel der Aufgabenbeschreibungen war auf ein Kärtchen geschrieben. Die Personen wurden gebeten, jedes Kärtchen auf einem dafür vorgesehenen

Platz abzulegen, sobald die entsprechende Formel ihrer Ansicht nach durch den aktuellen Netzentwurf erfüllt war. Außerdem wurden die Interaktionen eines Teils der Personen am Netzeditor per Video aufgezeichnet. Auf diese Weise konnten in der Analyse der Videoprotokolle die von den Personen durchgeführten Zuordnungen von Formelmengen zu Netzteilen ermittelt werden. Diese Zuordnungen führten zu den Designbausteinen. Abb. 4 zeigt drei der empirisch häufigeren Designbausteine. Dabei ist hervorzuheben, daß die meisten Designbausteine jeweils *einer* Formel ein Netzteil zuordnen (wie in Abb. 4), obwohl die Versuchsdurchführung auch die Zuordnung *mehrerer* Formeln zu Netzteilen gestattete.

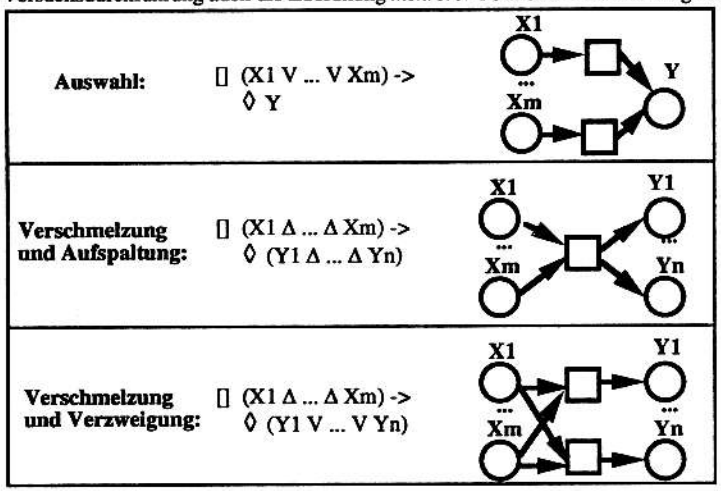


Abb. 4: Drei empirisch gewonnene heuristische Designbausteine

Der Lernende soll die Hilfen auf folgende Weisen benutzen können:

- Wenn der Lernende eine Designregel oder einen Designbaustein auswählt, dann soll sein aktueller Netzentwurf *automatisch* in entsprechender Weise *ergänzt* werden können.
- Wenn der Lernende seinen Netzentwurf oder Teile desselben vom System überprüfen lassen will, so kann er entsprechende *Prüfhypothesen* formulieren. Dazu markiert der Benutzer das zu überprüfende (Teil-) Netz. Er erhält dann vom System Rückmeldung über die aktuell erfüllten Formeln.

Dieser Entwurf des Hilfesystems soll in drei Richtungen erweitert werden:

- Basierend auf den Benutzerhandlungen soll ein *Benutzermodell* entwickelt werden, das den jeweils aktuellen Wissensstand des Benutzers repräsentiert. Vom Benutzermodell soll abhängen, welche Designregeln und Designbausteine dem Benutzer aktuell angeboten werden.
- Neben der direkten Überführung von Formeln in Netzteile soll auch die Ebene der Entwurfsplanung unterstützt werden. Für die Realisation von Konzepten wie Parallelität, wechselseitiger Ausschluß usw. sollen dem Benutzer entsprechende

Planbausteine zur Verfügung gestellt werden (s.a. [Dietz, Gronewold, Schreiber 91], Olderog 89].).

- Die Designregeln und Designbausteine des Systems sollen nicht fest eingegeben werden, sondern von dem System auf der Basis von Benutzerhandlungen *gelernt* werden.

Literatur:

[Anderson 89] Anderson, J.R., A Theory of the Origins of Human Knowledge, Artificial Intelligence, 1989, 40, 313-351

[Dietz, Gronewold, Schreiber 91] Dietz, Ch., Gronewald, A., Schreiber, G., Ein Übersetzungsverfahren von CCS-artigen Termen in Netze und deren Reduktion: Auswahl und Beschreibung der Verfahren, Bericht TI 6, FB Informatik, Theoretische Informatik, Universität Oldenburg, 1991

[Gollwitzer 90] Gollwitzer, P.M., Action Phases and Mind Sets: in Higgins, E.T., Sorrentino, R.M. (eds), Handbook of Motivation and Cognition: Foundations of Social Behavior, 1990, Vol. 2, 53-92

[Josko 90] Josko, B., Verifying the Correctness of AADL Modules using Model Checking. In: de Bakker, de Roever, Rozenberg (eds): Proceedings REX-Workshop on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness. Springer LNCS 430, 1990

[Möbus 91] Möbus, C., Wissenserwerb mit kooperativen Systemen, in Claus, V., Gorny, P., (Hg), Informatik: Wege zur Vielfalt beim Lehren und Lernen, Informatik-Fachberichte, Band 292, Berlin: Springer, 1991

[Möbus, Pitschke, Schröder 91a] Möbus, C., Pitschke, K., Schröder, O., Ein wissensstandsbezogenes Hilfesystem für Petrinetzmodellierer. In: Claus, V., Lichtblau, U., 1. Kolloquium der Arbeitsgruppe Informatik-Systeme AIS-1, 1991

[Möbus, Pitschke, Schröder 91b] Möbus, C., Pitschke, K., Schröder, O., Wissensstandsbezogenes Hilfesystem für Petrinetzmodellierer: Aufgabenanalyse und Instruktionsentwicklung. In: Claus, V., Lichtblau, U., 2. Kolloquium der Arbeitsgruppe Informatik-Systeme AIS-3, 1991

[Möbus, Pitschke, Schröder 91c] Möbus, C., Pitschke, K., Schröder, O., Towards the Theory-Guided Design of Help Systems for Programming and Modelling Tasks, to be presented at the Second International Conference on Intelligent Tutoring Systems ITS 92, June 10-12, 1992

[Möbus, Schröder, Thole 91] Möbus, C., Schröder, O., Thole, H.J., Runtime Modelling the Novice-Expert Shift in Programming Skills on a Rule-Schema-Case-Continuum, in Kay, J., Quilici, A. (eds), Proceedings of the IJCAI Workshop W.4 Agent Modelling for Intelligent Interaction, 12th Int. Joint Conf. on Artificial Intelligence, Darling Harbour, Sydney, Australia, 24 - 30 August, 1991

[Müller 88] Müller, H., Höhere Petri-Netze - ein Werkzeug für Modellierung und Analyse verteilter Systeme und paralleler Datenverarbeitungsprozesse, Informationstechnik, 1988, 30 (2), 110-117

[Olderog 89] Olderog, E.R., Nets, Terms, and Formulas: Three Views of Concurrent Processes and Their Relationship, Universität Oldenburg, Fachbereich Informatik, 1989

[vanLehn 88] vanLehn, K., Towards a Theory of Impasse-Driven Learning. In: Mandl, Lesgold (eds.) Learning Issues for Intelligent Tutoring Systems New York Springer 1988

[vanLehn 90] vanLehn, K., Mind Bugs: The Origins of Procedural Misconceptions. MIT Press 1990

[vanLehn 91] vanLehn, K., Rule Acquisition Events in the Discovery of Problem Solving Strategies, Cognitive Science, 1991, 15, 1-47

[Wolff 87] Wolff, J.G., Cognitive Development as Optimization, in Bolc, L. (ed), Computational Models of Learning, Berlin: Springer, 1987, 161-205