Unterlagen zur Vorlesung

Deduktionssysteme: Grundlagen und Anwendungen

Sommersemester 1999

Professor Dr. Manfred Schmidt-Schauß

Fachbereich Informatik
J.W.Goethe-Universität Frankfurt

1 Einführung und Überblick: Grundlagen und Anwendungen von Automatischen Deduktionssystemen

1.1 Literatur

Folgende Bücher und Handbuchartikel können als Grundlage für eine weiteres Studium dienen:

[And86,Duf91,Bib83,Bib92,BS92,BB87,RB79] [Bun83,CL73,EFT86,Gal86,GN87,DH89], [ki-98,Kow79,Lov78,Pau94,SA94,WOLB84,BS94,BS99]

Für spezieller Themenbereiche können auch folgende Bücher verwendet werden: [Ede92,FLTN93,Tha88,Smu71]

Literatur

[And:81] P. Andrews. Theorem proving via general matings. *J. ACM*, 28(2):193–214, 1981.

[And:86] Peter B. Andrews. *An Introduction to mathematical logic and type theory: to truth through proof.* Academic Press, 1986.

[BB:87] K.-H. Bläsisus and H.-J. Bürckert. *Deduktionssysteme - Automatisierung des logischen Denkens.* Oldenbourg Verlag, 1987.

[Bib:83] W. Bibel. *Automated Theorem Proving.* Vieweg, Braunschweig, 1983.

[Bib:92] W. Bibel. *Deduktion, Automatisierung der Logik.* Oldenbourg, 1992.

[BS:92] Wolfgang Bibel and Peter H. Schmitt, editors. volume I – III. Kluwer Academic Publishers, 1992.

[BS:94] Franz Baader and Jörg H. Siekmann. Unification theory. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 2. Oxford Science Publications, 1994.

[BS:99] Franz Baader and Wayne Snyder. Unification theory. In *Handbook of Automated Reasoning.* Elsevier Science Publishers, 1999. to appear.

[Bun:83] A. Bundy. *The Computer Modelling of Mathematical Reasoning.* Academic Press, London, 1983.

[CL:73] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, New York, 1973.

[DH:89] Ralf-Detlef Kutsche Dieter Hofbauer. *Grundlagen des maschinellen Beweisens.* Vieweg, 1989.

[Duf:91] David A. Duffy. *Principles of Automated Theorem Proving.* John Wiley & Sons, 1991.

[Ede:92] Elmar Eder. *Relative Complexities of First order Calculi.* Vieweg, Braunschweig, 1992.

[EFT:86] H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Einführung in die mathematische Logik.* Wissenschaftliche Buchgesellschaft Darmstadt, 1986.

[FLTN:93] C. Fermüller, A. Leitsch, Tammet T., and Zamov N. *Resolution methoda for the decision problem*, volume 679 of *LNCS.* Springer-Verlag, 1993.

[Gal:86] Jean H. Gallier. *Logical for Computer Science: Foundations of Automatic Theorem Proving.* Harper & Row Publishers, New York, 1986.

[GN:87] Michael R. Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann Publishers, 1987.

[Göd:31] K Gödel. über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Mh. Math. Phys*, 38:173–198, 1931.

[ki:98] Themenheft Deduktion und Anwendung, 1998. in german.

[Kow:79] R. Kowalski. *Logic for Problem Solving.* North Holland, Amsterdam, 1979.

[Lov:78] Donald W. Loveland. *Automated Theorem Proving: A Logical Basis*, volume 6 of *Fundamental Studies in Computer Science.* North-Holland, New York, 1978.

[Pau:94] Larry C. Paulson. *Isabelle: a Generic Theorem prover.* Springer-Verlag, 1994.

[RB:79] J S. Moore R. Boyer. *A Computational Logic*, volume 29. Academic Press, London, 1979.

[Rob:65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12:23–41, 1965.

[SA:94] Rolf Socher-Ambrosius. *Deduktionssysteme.* BI-Wissenschaftsverlag, 1994. in german.

[Smu:71] Raymond M. Smullyan. *First-Order Logic.* Springer-Verlag, 1971.

[Sti:86] Mark E. Stickel. A prolog technology theorem prover: implementation by an extended prolog compiler. In *Proc. of 8th Int. Conf. on Automated Deduction*, number 230 in Lecture Notes in Computer Science, pages 573–587. Springer-Verlag, 1986.

[Tar:53] A. Tarski. *Der Wahrheitsbegriff in den formalisierten Sprachen.* Studia Philospohica I. 1953.

[Tha:88] André Thayse, editor. *From Standard Logic to Logic Programming.* John Wiley & Sons, 1988.

[WOLB:84] L. Wos, R. Overbeek, E. Lusk, and J. Boyle. *Automated Reasoning – Introduction and Applications.* Prentice-Hall, Englewood Cliffs, NJ, 1984.

1.2 Vorbemerkung

Das Gebiet der Automatischen Deduktionssysteme ist relativ jung im Vergleich zum Gebiet der Logik.

Insbesondere bedeutet dies, dass Notationen und Begriffe sich z.T. nur im Englischen mit einer festen Bedeutung etabliert haben. Benutzt man Lehrbücher oder weiterführende Literatur (Handbücher, Artikel), so wird man mit verschiedenen Notationen und verschiedenen Namen für gleiche Begriffe konfrontiert. So werden manche englischen Begriffe auf verschiedene Weise ins deutsche übersetzt. Glücklicherweise sind die Begriffsbildungen diesselben, so daß beim Literaturstudium nach etwas Einarbeitung keine wesentlichen Schwierigkeiten auftauchen.

1.3 Behandelte Themen

Da diese Unterlagen mit der Vorlesung mitwachsen und nicht vorgefertigt vorliegen, hier ein vorläufiger Plan der Themen:

Prädikatenlogik erster Stufe
Resolution

- Klauselnormalformen und Resolution
- Unifikation
- Verfeinerungen der Resolution: Elimination von Redundanz.

– logisches Programmieren: SLD-Resolution

Analytische Tableaukalküle
Aussagenlogik Einfache automatische Beweismethoden.
Implementierte Deduktionssysteme
Gleichheitslogik Kalküle dazu.
Deduktionssysteme für Logik höherer Ordnung

1.4 Teilgebiete

Im folgenden eine Liste von Teilgebieten:

Beweiserbau Die Softwaretechnik der Automatischen Deduktionssysteme. Hier werden Fragen zur Softwaretechnik von Deduktionssystemen, effizienten Implementierungen, Datenstrukturen, Heuristiken untersucht. Ebenso Fragen der Benutzung: interaktiv, vollautomatisch, halbautomatisch, verifizierend, Erzeugung von lesbaren Beweisen.

Gleichheitsbehandlung Theorie und Verfahren zur Behandlung der Gleichheitsaxiomen. Für Herleitung reichen naiven Suchstrategien nicht aus. Aber: diese Relation kommt häufig in mathematischen Disziplinen und in der theoretischen Informatik vor.

Termersetzungssysteme Themengebiet sind gerichtete Gleichungen, deren Benutzung als Transformationssystem oder zum Entscheiden des Wortproblems. Kann man auch als Untergebiet der Gleichheitsbehandlung ansehen.

Unifikation Zentrale Fragestellung ist, ob zwei vorgegebene Strukturen mit offenen Stellen (Variablen) gleichgemacht werden können. Die Unifikationstheorie liefert spezielle Algorithmen und das theoretische Rüstzeug für die Untersuchung solcher Probleme. Diese spezielle Fragestellung tritt verbreiteter auf, als man annimmt: Frage der gemeinsamen instanzen von Pattern, Lösungen von Gleichungen, Logiken erster Ordnung, Gleichungstheorien, Lambda-Kalkül.

Nichtmonotone Logiken Eine Logik heißt monoton, wenn durch die Hinzunahme neuer Annahmen nicht weniger folgt als ohne diese Annahmen: $\Phi \models TH$ impliziert $\Phi \cup \Xi \models TH$. Die klassischerweise untersuchten Logiken sind monoton, viele Herleitungsprobleme der Künstlichen Intelligenz jedoch nicht.

Logische Methoden wissensbasierter Systeme Die wichtigsten Probleme sind die Entwicklung von Inferenzmethoden für wissensbasierte Systeme und die Mechanisierung nichtklassischer Logiken.

Modallogik Dies sind Gegenstand älterer logische/philosophische Untersuchungen. Erst durch den Versuch, geeignete Formulierungssprachen zur computergerechten Eingabe von verschiedenen Problemen zu finden, wurden diese Logiken wiederbelebt und für die Computerverarbeitung entdeckt.

Deduktive Programmsynthese Idee: Mit Hilfe deduktiver Verfahren ein Programm automatisch zu synthetisieren. Z.B.: In der konstruktiven Logik höherer Ordnung entsprechen Existenzbeweise einem ablauffähigen Programm.

Logisches Programmieren Die Prädikatenlogik, insbesondere das Fragment der Hornklauseln, wird als (Grundlage einer) Programmiersprache (Prolog) aufgefaßt. Implementierungen, Erweiterungen (Negation, Constraints, ..), werden untersucht.

Anwendungen Automatischer Deduktionssysteme

- logische Programmiersprachen, Constraints + logische Programmiersprachen.
- deduktiven Datenbanken
- Planen, Plangenerierung in Systemen der KI.
- Diagnose.
- Nachweis der Korrektheit von Hard- oder Software:
  - Programmverifikation
  - Programmanalyse
  - Untersuchung von Sicherheitsprotokollen
- Inferenzkomponenten für wissensbasierte Systeme
- Wissensrepräsentationformalismen

Wichtigste Methoden

Resolution
Tableaukalküle
Gleichheit Termersetzung, Paramodulation, Superposition
Modelchecking

1.5 Einige computergeführte Beweise

Einige lange Zeit offene Problemen, die mit Computerhilfe gelöst wurden, (allerdings nicht mit allgemeinen automatischen Deduktionssystemen), sind:

- **Der Vierfarbensatz:**
  jede ebene Landkarte läßt sich mit vier Farben färben. Die Lösungsmethode war ein spezielles Programm, das einige tausend Konfigurationen zu untersuchen hatte.

- **Nichtexistenz einer endlichen projektiven Ebene der Ordnung 10.** Hintergrund ist die Axiomatisierung der projektiven Ebenen:

  1. Durch je zwei Punkte geht genau eine Gerade.
  2. Je zwei Geraden schneiden sich in einem Punkt.
  3. Es gibt 4 verschiedene Punkte von denen keine drei kollinear sind.

  Stichwortartig einige Eigenschaften: Ordnung sei n; n + 1 = Anzahl der Punkte auf einer Geraden = Anzahl der Geraden durch einen Punkt. Alle Geraden haben gleichviele Punkte $(n + 1)$. Durch alle Punkte gehen gleich viele Geraden $(n+1)$. Anzahl aller Punkte = Anzahl aller Geraden = n^2+n+1 Man kennt zu jeder Primzahlpotenz p^n (mindestens) eine projektive Ebene dieser Ordnung. Aber: Man weiß nicht, ob es eine endliche projektive Ebene zu einer Ordnung gibt, die keine Primzahlpotenz ist. Die kleinste Ordnung für die dies lange Zeit unbekannt war, ist 10. Die nächste offene Ordnung ist 12.
- Die "Kepler-Vermutung": Diese Vermutung ist (war) bereits über dreihundert Jahre alt.

  Sie besteht darin, daß die dichteste Kugelpackung im Raum die ist, die man durch Probieren sofort findet:

  Nämlich die Art der Stapelung von Melonen oder von Kanonenkugeln, die zu einer Pyramide führt: In der ersten Ebene sind die Kugeln wabenförmig aneinandergelegt. Die nächste Ebene sieht genauso aus, nur leicht versetzt, so daß die Kugeln der zweiten Ebene in die Täler der ersten Ebene fallen, usw. In diesem Fall werden $\frac{\pi}{\sqrt{18}} \sim 0,74048$ des Raumes von Kugeln gefüllt.

  Der mathematische Fallstrick ist offenbar, daß die lokale Minimierung des Volumens, das von einer Kugel und den (maximal) 12 umgebenden Kugeln benötigt wird, eine scheinbar bessere Packung ergibt; allerdings läßt sich diese nicht fortsetzen.

  Die Kepler-Vermutung wurde gelöst von Tom Hales und seinem Doktoranden Sam Ferguson (University of Michigan) (weitere Info unter `http://www.math.lsa.umich.edu/ hales`, siehe ebenso einen aktuellen "Zeit"-Artikel)

  Die Lösung erforderte umfangreiche mathematische Vorarbeit, bis man das Problem darauf reduziert hat, Kugelhaufen mit 53 Kugeln zu untersuchen. Hiervon gibt es 5000 Typen, die alle untersucht werden müssen. Dies wurde mit Hilfe eines Computers durchgeführt.

Bei diesen Problemen wurden spezialisierte Systeme entwickelt, die eine Art logisches Numbercrunching durchgeführt haben. Beim Vierfarbensatz und beim Kepler-Problem nach einer erheblichen mathematischen Vorarbeit, beim projektiven-Ebenen-Problem-10 ist es geordnetes, optimiertes Durchprobieren von sehr vielen Möglichkeiten bei der Suche nach einem Modell.

Automatische Deduktionssysteme, die allgemeiner konstruiert sind, haben auch schon Spitzenleistungen erzielt: z.B. Lösung des sogenannten Problems der Robbins-Algebren:

1.6 Das Problem der Robbins Algebren

Das Robbins Problem “sind alle Robbins-Algebren auch Boolesche Algebren?“ wurde von William McCune, Automated Deduction Group, Mathematics and Computer Science Division, Argonne National Laboratory gelöst und zwar mit Hilfe des automatischen Beweisers EQP für Gleichungstheorien. Die Lösung ist:

Jede Robbins-Algebra ist eine Boolesche Algebra.

Zunächst: Eine Boolesche Algebra hat zwei zweistellige, kommutative, assoziative Operatoren `und, oder`, einen einstelligen Operator `not`, Konstanten 0,1, so daß folgendes gilt: die De-Morganschen Gesetze, Distributivität, `not` ist ein Komplementoperator: x `und` (`not` x) $= 0$, x `oder` (`not` x) $= 1$, `not`(`not` x) $= x$.

Etwas Geschichte:

Im Jahre 1933 gab E. V. Huntington die folgende (alternative) Axiomatisierung einer Booleschen Algebra:

$$
\begin{array}{lr}
x + y = y + x & \text{[kommutativ]} \\
(x + y) + z = x + (y + z) & \text{[assoziativ]} \\
n(n(x) + y) + n(n(x) + n(y)) = x & \text{[Huntington Gleichung]}
\end{array}
$$

In dieser Axiomatisierung kann man `und` mittels `oder` und `not` definieren, und nachweisen, dass alle Axiome einer Booleschen Algebra dafür gelten.

Danach vermutete Robbins, dass man das letzte Axiom durch eine anderes ersetzen kann:

$$n(n(x + y) + n(x + n(y))) = x \text{ [Robbins equation]}$$

Robbins und Huntington konnten keinen Beweis finden, später arbeitete Tarski mit Studenten an diesem Problem, ebenfalls ohne Erfolg.

Die Algebren mit den Axiomen Kommutativität, Assoziativität und Robbins Axiom wurden bekannt als Robbins Algebren. Man sieht leicht, dass Boolesche Algebren auch Robbins Algebren sind; die interessante offene Frage war die Umkehrung.

Seit 1979 arbeitete daran: Steve Winker, Larry Wos mit dem Beweiser “Otter“ Die Methode war schrittweise immer stärkere Bedingungen (d.h. zusätzliche Axiome) zu finden, unter denen eine Robbins Algebra auf jeden Fall auch eine Boolesche Algebra ist. Hierbei wurden die Beweise vollautomatisch gesucht. Winker fand viele solche Bedingungen und bewies dies unter Benutzung von Otter : U.a.:

$$
\begin{array}{ll}
\forall x : n(n(x)) = x & \\
\exists 0 : \forall x : x + 0 = x & \\
\forall x : x + x = x & \\
\exists C : \exists D : C + D = C & \text{[erste Winker-Bedingung, 1992]} \\
\exists C : \exists D : n(C + D) = n(C) & \text{[zweite Winker-Bedingung,1996]}
\end{array}
$$

Der automatische Beweis der Lösung des Robbins Problem wurde am 10 Oktober 1996 mittels EQP geführt. EQP benutzt assoziative-kommutative Unifikation und kann nur Gleichheitslogik.
Die Gleichungen

$$n(n(n(y)+x)+n(x+y)) = x \qquad \text{[Robbins Axiom]}$$
$$n(x+y) \neq n(x) \qquad \text{[Negation von Winker-2]}$$

wurden von EQP zum Widerspruch geführt. Die Suche dauerte 8 Tage. Ein Problem war die Prüfung des von EQP gefundenen Beweises. Hierzu mußte mittels Otter ein (maschinen-)lesbarer Beweis erzeugt werden, der dann wieder von einem anderen Beweisprüfprogramm nochmal geprüft wurde.

Für weitere Informationen, siehe die zugehörigen Web-Seiten:

```
www-unix.mcs.anl.gov/ mccune/papers/robbins/lemmas.html
www.mcs.anl.gov/AR/eqp
www.mcs.anl.gov/AR/otter
```

1.7 Methodenwahl: menschliche oder maschinelle?

Was wird erfolgreicher sein?

- Die Nachahmung der menschlichen Vorgehensweise, oder
- Die (intelligente) brute-force (numbercrunching)-Methode

Die bisherige Erfahrung zeigt, daß, (wenn man einmal vom Einsatz eines Rechners wie beim 4-Farben-Problem absieht), der Einsatz von Computern erfolgreicher ist, wenn man nicht das menschliche Problemlösungsverhalten nachahmt, sondern die zweite Variante, die (intelligente) brute-force Methode verwendet. Z. B. sind die Erfolge der Schachprogrammierung eher auf solche Methoden als auf rational planende und nachdenkende Methoden zurückzuführen.

Genauso verhält es sich z.B. mit der Lösung des Robbins-Algebra-Problems. Wobei man hier einwenden könnte, daß ohne einen Menschen, der das System EQP intelligent einsetzt, der Beweis nicht gefunden worden wäre.

Es gibt einen Zweig des “Automated Reasoning“, der gute Gründe dafür anführt, daß eine Analogie zum menschlichen Planen notwendig sei; allerdings sind die großen Erfolge hier bisher ausgeblieben.

1.8 Einführende Beispiele

Aussagen über die reale Welt oder über fiktive oder abstrakte Welten sind manchmal nicht unabhängig voneinander, sondern es kann der Fall sein, daß eine Aussage aus einer oder mehreren anderen Aussage folgt. Zum Beispiel würde man bei den Aussagen

“Jede Katze ißt Fische“
“Garfield ist eine Katze“
“Garfield ißt Fische“

sicherlich zustimmen, daß die dritte Aussage aus den beiden anderen folgt. Wann immer man annimmt, daß die ersten beiden Aussagen wahr sind, muß man notwendigerweise auch annehmen, daß die dritte Aussage wahr ist. Das gleiche gilt für folgende Aussagen

"Alle Menschen sind sterblich"
"Sokrates ist ein Mensch"
"Sokrates ist sterblich".

Wieder gilt, daß die dritte Aussage aus den ersten beiden folgt. Das interessante dabei ist, daß der Grund weshalb die dritte Aussage aus den beiden anderen folgt offensichtlich in beiden Fällen derselbe ist. Es hängt gar nicht davon ab, ob wir über Fische essende Katzen oder sterbliche Menschen oder über irgendwelche unbekannten Objekte mit unbekannten Eigenschaften sprechen:

"Alle X haben die Eigenschaft E"
"A ist ein X"
"A hat die Eigenschaft E"

Offensichtlich folgt die dritte Aussage aus den beiden anderen. Die Beobachtung, daß die "folgt aus" Beziehung zwischen verschiedenen Aussagen gar nicht notwendigerweise aus deren Bedeutung definiert werden muß, sondern nur aus deren syntaktischer Form geht zurück auf Aristoteles und andere antike griechische Philosophen. Die verblüffende Konsequenz ist, daß eine an und für sich semantische, d.h. von der Bedeutung her, definierte Beziehung, syntaktisch, d.h. über rein mechanische Symbolmanipulation, realisiert werden kann. In diesem Fall kann der Test auf die Folgerungsbeziehung zwischen Aussagen auch Computerprogrammen übertragen werden. Programme mit diesen Fähigkeiten heißen **Deduktionssysteme**. Unter einem Deduktionssystem kann man daher ganz allgemein ein programmierbares Verfahren verstehen, das aus vorhandenen Daten Schlußfolgerungen zieht und damit neue Daten ableitet. Welche Art von Schlußfolgerungen gemeint ist, illustriert "Modus Ponens", eine typische Regel, der auch die beiden obigen Beispiele zum Teil zugrundeliegen, und die schon auf Aristoteles zurückgeht.

Fakt 1:	P	(P und Q sind beliebige Aussagen.)
Fakt 2:	Aus P folgt Q	
Schlußfolgerung:	Q	

Die Mechanismen, die einen Rechner zu solchen Ableitungen befähigen, sind im Prinzip sehr ähnlich zu denen, die ihn zum Rechnen mit Zahlen befähigen. Die Fakten müssen als Datenobjekte repräsentiert und die Schlußfolgerungen als Operationen auf diesen Datenobjekten programmiert werden.

1.9 Motivationsbeispiele

Um einen allerersten Eindruck von der Arbeitsweise eines Deduktionssystems zu bekommen, sollen zwei Beispiele zu deren Anwendung vorgestellt werden. Das erste ist ist das Sokrates-Beispiel, und das zweite ist ein logisches Rätsel. Die beiden Beispiele sollen nur einen ungefähren Eindruck geben und demonstrieren, daß man offensichtlich nicht nur Fische fressende Garfields oder sterbliche Philosophen, sondern wirklich nichttriviale logische Zusammenhänge mit automatisierbaren Methoden behandeln kann.

Beispiel 1.1. Wir wollen beweisen: Aus

"Alle Menschen sind sterblich"
"Sokrates ist ein Mensch"

folgt

"Sokrates ist sterblich".

Zunächst benötigt man eine formale Sprache, in der man diesen Sachverhalt ausdrücken kann. Dafür bietet sich in diesem Fall die Prädikatenlogik an. Das ist die formale Sprache, in der die meisten mathematischen Zusammenhänge einfach aufgeschrieben werden können, und die daher den meisten Lesern sowieso mehr oder weniger vertraut sein sollte. Da es in diesem Beispiel um zwei Eigenschaften geht, führen wir `Mensch` und `sterblich` als Prädikatensymbole ein. `Sokrates` ist ein Konstantensymbol in der Sprache. Die drei Aussagen sind dann:

1.) $\forall x : \texttt{Mensch}(x) \Rightarrow \texttt{sterblich}(x)$
2.) $\texttt{Mensch}(\texttt{Sokrates})$
3.) $\texttt{sterblich}(\texttt{Sokrates})$

Der herzuleitende Aussage wird jetzt, um einen Beweis durch Widerspruch zu demonstrieren, verneint. Damit hat man eine Menge von Aussagen, auf die man das mechanische Verfahren der Resolution anwenden kann. Das Ziel ist es einen Widerspruch herzuleiten. Hat man dies erreicht, dann kann man sagen, daß die verneinte Aussage aus den anderen folgt.

1.) $\forall x : \texttt{Mensch}(x) \Rightarrow \texttt{sterblich}(x)$
2.) $\texttt{Mensch}(\texttt{Sokrates})$
3.) $\neg\texttt{sterblich}(\texttt{Sokrates})$

Zunächst muß man die Formeln noch etwas abändern:

1.) $\forall x : \neg\texttt{Mensch}(x) \vee \texttt{sterblich}(x)$
2.) $\texttt{Mensch}(\texttt{Sokrates})$
3.) $\neg\texttt{sterblich}(\texttt{Sokrates})$

Das mechanische Verfahren der Resolution erlaubt es jetzt, aus 1) und 3) durch instantiieren von x mit `Sokrates` die Formel

$\neg\texttt{Mensch}(\texttt{Sokrates})$.

herzuleiten. Diese Formel und die zweite erlauben es jetzt, den endgültigen Widerspruch direkt zu erkennen.

Beispiel 1.2. **Wise Men Puzzle** Es war einmal ein König, der hatte drei weise Männer. Eines Tages beschloss er, herauszufinden, wer von den dreien der weiseste ist. Er stellt sie einander gegenüber, macht jedem von ihnen einen weißen oder schwarzen Punkt auf die Stirn und sagt ihnen, daß mindestens einer der Punkte weiß ist. Nach einer Weile, in der niemand was sagte, fragte er den ersten, ob er die Farbe seines Punktes wüßte. Der verneinte. Dann fragte er den zweiten. Der verneinte ebenfalls. Daraufhin sagte der dritte prompt, daß sein Punkt weiß ist. Woher wußte der das?

Zur Lösung dieses Rätsels ist es notwendig, Schlüsse über den Wissensstand der Beteiligten zu ziehen. Zur Formulierung dieses Rätsels ist die prädikatenlogische Sprache daher nicht sehr gut geeignet. Einfacher wird es, wenn man die logische Sprache um neue Operatoren erweitert, mit denen man ausdrücken kann, daß jemand eine bestimmte Sache weiß oder nicht weiß. Damit macht man aus der Prädikatenlogik eine sogenannte *epistemische Logik*. Dies führt zur sogenannten Modallogik und deren Varianten. Es gibt Methoden, die Aussagen dieser Logik wieder in Prädikatenlogik übersetzen.