3 Überblick über den Aufbau und Funktionsweise eines Automatischen Deduktionssystems am Beispiel eines Resolutionsbeweisers

Ein implementiertes Automatisches Deduktionssystem baut sich im wesentlichen aus vier Schichten auf – Logik, Kalkül, logisches Zustandsübergangssystem und Steuerung. Im folgenden sollen die Ideen informell skizziert werden. Später werden diese in Definitionen präzisiert.

Logik Die unterste Schicht eines Deduktionssystems wird durch eine Logik gebildet, die mit der Festlegung der Syntax einer formalen Sprache und deren Semantik die zulässige Struktur und die Bedeutung von Aussagen vorgibt. Aussagen entsprechen Formeln der Logik. Die gewählte Logik bestimmt ganz konkret, welche Arten von Aussagen erlaubt und welche verboten sind. Beispielsweise kann sie festlegen, daß eine Quantifizierung "Für alle Zahlen gilt ..." erlaubt, eine Quantifizierung "Für alle Funktionen über natürlichen Zahlen gilt ..." dagegen verboten sein soll. Die Definition einer Bedeutung (Semantik) für die Formeln liefert darüberhinaus eine Beziehung zwischen Aussagen, die als "aus A folgt B" gedeutet werden kann. Damit ist ein *semantischer Folgerungsbegriff* etabliert, der zunächst jedoch in keiner Weise hilft, für gegebene A und B algorithmisch zu bestimmen, ob B wirklich aus A folgt. Wie in den Motivationsbeispielen schon angedeutet wurde, gibt es jedoch nicht die eine Logik, sondern ähnlich wie bei Programmiersprachen gibt es eine ganze Hierarchie von Logiken mit vielen unterschiedlichen Varianten und Erweiterungen nach verschiedenen Richtungen. Jede von ihnen formalisiert ganz bestimmte Grundkonzepte, auch wiederum ähnlich wie in Programmiersprachen. Genauso wie man sich für normale Anwendungen eine möglichst gut geeignete Programmiersprache aussucht, muß man für konkrete deduktive Anwendungen eine **adäquate Logik** auswählen. Für einfache Puzzles kann die Aussagenlogik ausreichen, während für kompliziertere Anwendungen und Beispiele (wise-man puzzle) eine Multi-Modallogik gut geeignet ist. Eng mit der Auswahl der Logik verknüpft ist die Frage der Formulierung eines Problems, d.h. "wie formuliere ich das Problem in der jeweiligen Logik am besten?". Z. B. hätte man das wise-man puzzle auch rein in Prädikatenlogik formulieren können. Offensichtlich ist diese Formulierung schwieriger. Damit stellt sich auch die Frage; "ist das Problem überhaupt richtig formuliert ?", also nach der "Korrektheit" der Formulierung.

Die elementarste Logik ist die *Aussagenlogik*. Sie formalisiert die Grundkonzepte *wahr* und *falsch* und führt darauf die Booleschen Verknüpfungen $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, \ldots$ ein. In Aussagenlogik kann man z.B. " es-regnet $\Rightarrow$ Straße-wird-nass" hinschreiben, aber nicht über irgendwelche Mengen quantifizieren. Das Problem, herauszufinden, ob eine Aussage aus einer anderen folgt ist in Aussagenlogik entscheidbar.

Das gilt jedoch nicht mehr in der nächstkomplizierten Logik, der *Gleichungslogik*. Neben Quantifizierungen über Mengen führt Gleichungslogik das Konzept der Funktion ein. Man kann dann z.B. hinschreiben:

$$\forall x, y: \ x + 0 = x \wedge x + s(y) = s(x + y)$$

Die Formel beschreibt die Addition auf natürlichen Zahlen wobei $s(x)$ als $x + 1$ zu interpretieren ist. Man muß aber beachten, daß s eine syntaktische Repräsentation der +1-Funktion ist, und nicht die Funktion selbst. Betrachtet man die tatsächlichen Zahlen als Abkürzungen für entsprechend viele Verschachtelungen der Funktion s, dann kann man mit Hilfe dieser Axiome schon richtig rechnen. Zum Beispiel ergibt sich:

$$3 + 2 = s(s(s(0))) + s(s(0)) = s(3 + s(0)) = s(s(3 + 0)) = s(s(3)) = 5$$

Gleichungslogik ist aber wiederum nur ein Spezialfall von *Prädikatenlogik*, wo neben der ganz speziellen Relation der Gleichheit ganz allgemeine Relationen als neues Konzept hinzukommen. Hier kann man dann beispielsweise formulieren:

$$\forall x, y: \ Katze(x) \wedge Vogel(y) \Rightarrow Mag(x, y) Katze(Garfield) \wedge Vogel(Tweety)$$

"Katze" und "Vogel" bezeichnen einstellige Relationen, die auf ein Objekt zutreffen können oder nicht. "Mag" bezeichnet eine zweistellige Relation. In der Prädikatenlogik gibt es Relationen mit beliebiger Stelligkeit.

Eine weitere Logik ist zum Beispiel die *Modallogik*, die man zur (automatischen) Lösung des Wise Men Puzzles benutzen kann. Das Grundkonzept, das bei den Modallogiken hinzukommt, ist das der Zustände – manchmal auch Welten genannt – und der Zustandsübergänge. Wie beim Wise Men Puzzle macht es offensichtlich Sinn, nicht nur über eine Welt zu reden, in der eine Aussage einen bestimmten Wahrheitswert hat, sondern es kann notwendig sein, gleichzeitig über viele verschieden hypothetische Welten und deren Beziehungen zueinander zu reden. Genau das ist mit Modaloperatoren in sehr eleganter Weise möglich. Beim wise-man-puzzle kann man sich vorstellen, dass jeder ein eigene Menge von gültigen Aussagen haben kann.

Von Logikern und Philosophen wurden noch eine Unzahl anderer Logiken und deren Beziehungen zueinander untersucht. Für eine konkrete Anwendung ist die Situation daher wie bei der Wahl einer geeigneten Programmiersprache. Die beste ist die, die das, was in der Anwendung vorkommt möglichst einfach und elegant zu formulieren erlaubt, und für die es einen möglichst guten Compiler oder Interpretierer gibt. Die Formalismen, die man braucht, um eine Logik zu beschreiben und zu untersuchen sind zunächst eine Beschreibung der Syntax, d.h. der Grammatik der Sprache und eine Beschreibung der Bedeutung der syntaktischen Elemente, d.h. der Semantik. Der Semantikformalismus, der uns hier interessiert, geht zurück auf Alfred Tarski und wird deshalb auch Tarski-Semantik genannt. Die Idee ist, eine Abbildung der syntaktischen Elemente auf mathematische Objekte anzugeben, und diese Abbildung zu benutzen, um dann Terme auf Objekte einer bestimmten Menge, der. sogenannten *Trägermenge* oder auch *Universum* genannt, abzubilden, und Formeln zu wahr oder falsch zu evaluieren. Eine solche Abbildung heißt auch *Interpretation.* Typischerweise bildet man in diesem Formalismus Konstanten- und Variablensymbole auf Elemente der Trägermenge, Funktionssymbole auf Funktionen und Prädikatensymbole auf Relationen ab. Eine solche Abbildung, die eine Formel wahr macht heißt *Modell* der Formel. Zum Beispiel kann man für die Formel

$$Katze(Garfield) \wedge Vogel(Tweety)$$

als Trägermenge tatsächlich die Menge aller Katzen und Vögel nehmen. Das Prädikatensymbol "Katze" wird auf die Relation $Katze$ abgebildet, d.h. $Katze(x)$ trifft zu wenn x wirklich eine Katze ist. Entsprechend machen wir das mit dem Symbol "Vogel". Wenn jetzt die Konstantensymbole "Garfield" und "Tweety" tatsächlich auf eine Katze bzw. Vogel abgebildet werden, dann ist die Formel wahr und wir haben damit ein Modell der Formel. Wenn wir stattdessen das Symbol "Garfield" auf einen Vogel abbilden, dann ist die Formel falsch und die Interpretation ist kein Modell. Genausogut können wir auch als Trägermenge die natürlichen Zahlen nehmen und das Prädikatensymbol "Katze" auf die Relation $gerade$ und das Prädikatensymbol "Vogel" auf die Relation $ungerade$ abbilden. Wenn jetzt z.B. "Garfield" auf die 0 und "Tweety" auf die 1 abgebildet wird haben wir wieder ein Modell der Formel. Man bezeichnet eine Formel als allgemeingültig oder als *Tautologie* wenn sie unter *allen* Interpretationen wahr ist. Typische Tautologien sind "$P \vee \neg P$", aber auch Formeln wie "Euklidische Geometrieaxiome $\Rightarrow$ Satz des Pythagoras". Formeln, die unter keiner Interpretation wahr sind heißen *unerfüllbar* oder *widersprüchlich*. Typische Widersprüche sind "$P \wedge \neg P$", aber auch Formeln wie "$\neg$ (Euklidische Geometrieaxiome $\Rightarrow$ Satz des Pythagoras)". Formeln, die weder allgemeingültig noch widersprüchlich sind, die man also je nach Interpretation wahr oder falsch machen kann, heißen *erfüllbar*. Die obige Formel $Katze(Garfield) \wedge Vogel(Tweety)$ ist von diesem Typ. Die Tarski Semantik erlaubt die Definition eines semantischen Folgerungsbegriffs: $F \models G$ (G folgt aus F) falls G in allen Modellen von F gilt. Das heißt, wie immer auch die Symbole, die in F vorkommen interpretiert werden, falls F unter dieser Interpretation wahr ist, muß auch G unter dieser Interpretation wahr sein. Dieser semantische Folgerungsbegriff definiert präzise, was "Folgerung" heißen soll. Da die Anzahl der möglichen Interpretationen meist unendlich ist, gibt sie aber keinen Hinweis, wie man die Folgerungsbeziehung für zwei konkrete Formeln F und G auch tatsächlich überprüfen kann (das ist Sache der zweiten Stufe eines Deduktionssystems, des Kalküls). Spezielle Fragestellungen, die man auf der Ebene der Logiken noch untersucht, sind z.B. die Beziehung zwischen semantischer Folgerung und syntaktischer Implikation, d.h. die Frage, ob das sogenannte *Deduktionstheorem* gilt:

$F \models G$ gdw. $F \Rightarrow G$ allgemeingültig.

In Prädikatenlogik gilt es, in anderen Logiken nicht unbedingt. Eine andere Fragestellung ist mehr anwendungsorientiert, nämlich die Frage nach der Existenz von Übersetzern, die Formeln einer Logik in eine andere übersetzen. Analog wie bei der Entwicklung einer Programmiersprache ein Compiler die Ausführung von Programmen erheblich beschleunigen kann, kann ein Übersetzer zwischen Logiken den Test der Folgerungsbeziehung erheblich erleichtern. Beispiele für solche

Übersetzer sind die Transformation von epistemischer Logik in Prädikatenlogik, die bei der Lösung des Wise Men Puzzles benutzt werden kann. Ein weiteres Beispiel ist die Transformation in Klauselnormalform. Klauselnormalform ist eine echte Unterklasse der Prädikatenlogik, so daß man diese Transformation auch als Übersetzung in eine andere Logik sehen kann.

Kalkül Der in der Logik definierte semantische Folgerungsbegriff hilft meist nicht, für gegebene A und B algorithmisch zu bestimmen, ob B wirklich aus A folgt. Dies ist Aufgabe des *Kalküls*, der zweiten Schicht eines Deduktionssystems. Ein Kalkül definiert syntaktische *Ableitungen* als Manipulationen auf den (syntaktisch gegebenen) Formeln. Damit kann aus einer Formel A durch reine Symbolmanipulation eine Formel B gewonnen werden, wobei die Bedeutung der in A und B vorkommenden Symbole überhaupt keine Rolle spielt. Ein syntaktisch aus A abgeleitetes B soll aber trotzdem semantisch folgen und umgekehrt. Ein korrekter Kalkül stellt daher nur solche Ableitungsoperationen zur Verfügung, die garantieren, daß alles syntaktisch Ableitbare auch semantisch folgt. Wenn umgekehrt alles, was semantisch folgt, auch syntaktisch ableitbar ist, ist der Kalkül vollständig. Es gibt auch Kalküle, die statt auf Formeln auf anderen Datenstrukturen operieren, z.B. i) Folgen von Formeln (Sequenzenkalkül) , ii) Mengen von Formeln, iii) auf einem Graph, der u.a. mit Formeln markiert ist.

Nach dem Unvollständigkeitssatz von Kurt Gödel [Göd31] sind vollständige Kalküle ab einer gewissen Ausdrucksstärke der Logiken jedoch nicht möglich. Dazu gehört die sogenannte Prädikatenlogik zweiter Stufe, in der neben Quantifizierungen über Mengen auch Quantifizierungen über Funktionen über den Mengen erlaubt sind. Hierzu zählt auch die Logik, die die Theorie der natürlichen Zahlen beschreibt, d.h. alle Formeln die genau für die natürlichen Zahlen gelten. In diesen Logiken gibt es Aussagen, die über den semantischen Folgerungsbegriff aus anderen folgen, was aber mit keinem Kalkül, der nur mit Symbolmanipulation arbeitet, nachgewiesen werden kann.

Meist werden Ableitungsregeln folgendermaßen geschrieben: $\dfrac{F_1, \ldots, F_n}{F}$. Das soll bedeuten, wenn $F_1, \ldots, F_n$ schon abgeleitete Formeln sind, dann ist es auch erlaubt, die Formel F abzuleiten. Eine typische Ableitungsregel dieser Art ist die *Instantiierungsregel:*

$$\frac{\forall x:\ F[x]}{F[t/x]}$$

Dabei ist $F[x]$ eine Formel, in der die Variable x vorkommt und $F[t/x]$ ist eine Variante von F, bei der alle Vorkommnisse von x durch den Term t ersetzt worden sind (t ist beliebig). Die Regel besagt, daß eine Aussage, die für alles gilt ($\forall x$) auch für jedes spezielle Objekt gilt. Zweimal angewendet läßt sich damit zum Beispiel aus

$$\forall x, y:\ Katze(x) \land Vogel(y) \Rightarrow Mag(x, y)$$

ableiten:

$$Katze(Garfield) \wedge Vogel(Tweety) \Rightarrow Mag(Garfield, Tweety)$$

Eine weitere Regel dieser Art ist die *Modus Ponens* Regel, die schon auf die griechischen Philosophen zurückgeht:

$$\begin{array}{l} A \Rightarrow B \\ A \\ \hline B \end{array}$$

Sie drückt folgendes aus: wenn es gilt, daß aus der Aussage A die Aussage B folgt und wenn weiterhin bekannt ist, daß die Aussage A tatsächlich wahr ist, dann darf man annehmen, daß auch die Aussage B wahr ist.

Beide Regeln, die Instantiierungsregel als auch die Modus Ponens Regel eignen sich als reine Zeichenkettenmanipulationen und lassen sich daher problemlos auf einem Rechner ausführen.
Typische Fragestellungen auf der Kalkülebene sind folgende:

Korrektheit eines gegebenen Kalküls: Folgt alles abgeleitete auch semantisch?
Vollständigkeit eines gegebenen Kalküls: Läßt sich alles semantisch folgerbare auch ableiten?

Für die Praxis hat sich gezeigt, daß die Vollständigkeit so wie oben definiert an den Kalkül leicht angepasst werden muß. Was man z.B. bei Resolutionsbeweisern wirklich braucht ist die *Widerlegungsvollständigkeit*: Wenn aus einer Aussage F eine Aussage G semantisch folgt, ist dann ein Widerlegungsbeweis für $F \wedge \neg G$ möglich? Resolution ist z.B. ein Kalkül, der nicht vollständig , aber widerlegungsvollständig ist. Beispielsweise folgt aus A die Aussage $A \vee B$. Sie ist aber nicht mit Resolution ableitbar. Jedoch ist $A \wedge \neg(A \vee B)(= A \wedge \neg A \wedge \neg B)$ mit Resolution widerlegbar. Eine weiteres Problem im Bereich der Kalküle ist die Frage nach der *Effizienz* eines Kalküls. Dabei betrachtet man meist zwei Effizienzkriterien, i) die Verzweigungsrate im Suchraum und die ii) Länge der Beweise. Die Regeln eines Kalküls sind im allgemeinen an vielen verschiedenen Stellen einer Formelmenge anwendbar. Nicht alles, was damit abgeleitet wird, ist jedoch für den gesuchten Beweis brauchbar. Daher definieren die Regeln eines Kalküls einen Suchraum, der durch irgendein Suchverfahren abgesucht werden muß. Je größer die Verzweigungsrate, d.h. je mehr Stellen es in der aktuellen Formelmenge gibt, auf die die Kalkülregeln anwendbar sind, desto aufwendiger ist meist die Suche. Die Verzweigungsrate ist jedoch kein generelles Kriterium. Ist die Verzweigungsrate niedrig, liegt aber dafür der gesuchte Beweis sehr tief im Suchraum, dann ist auch nicht viel gewonnen. Was man braucht ist einerseits eine möglichst kleine Verzweigungsrate und andererseits Kalkülregeln, die einen Beweis mit möglichst wenig Schritten finden, und diese Schritte sollen mit möglichst wenig Aufwand berechenbar sein. Die Kunst des Kalkülentwerfens besteht darin, solche Kalküle zu entwickeln, die einen guten Kompromiss zwischen allen drei Faktoren bilden.

Die rein syntaktisch arbeitenden Kalküle wie Resolution, die nichts von der Bedeutung der Symbole, die sie manipulieren wissen, sind zwar im Prinzip ausreichend. Für viele häufig vorkommende Konstrukte kennt man jedoch Algorithmen und Kalküle, die spezielle Probleme erheblich schneller und besser lösen als die universell anwendbaren Kalküle. Niemand würde zum Beispiel auf die Idee kommen, die Gleichung $3 + 4 = x$ mit einem universellen Kalkül wie Resolution zu lösen. Das rechnet man einfach aus. Daher besteht ein weiterer Zweig der Kalkülentwicklung darin, Algorithmen für wichtige Spezialfälle zu integrieren, bzw. erst überhaupt zu entwickeln. Als ein Rahmenkonzept dafür hat sich die von Mark Stickel vorgeschlagene *Theorieresolution* [Sti86] als sehr nützlich gezeigt. Die Idee dabei ist, einen Resolutionsschritt nicht mehr über die syntaktische Komplementarität – gleiches Prädikatensymbol und Argumente, verschiedenes Vorzeichen – sondern über semantische Widersprüchlichkeit der Resolutionsliterale zu steuern. Diese semantische Widersprüchlichkeit kann von einem speziellen Algorithmus, der über die Bedeutung der vorkommenden Symbole etwas weiß, getestet werden. Beispielsweise kann man damit folgern:

$$\begin{array}{l} a < b \lor P \\ a > b \lor Q \\ \hline P \lor Q \end{array}$$

eben weil $a < b$ und $a > b$ unter der üblichen Bedeutung von $<$ und $>$ widersprüchlich sind. Ein Teil des Skripts ist solchen Spezialverfahren gewidmet.

Logische Zustandsübergangssysteme Die dritte Schicht eines Deduktionssystems, die Schicht der logischen Zustandsübergangssysteme, bestimmt die Darstellung von Formeln oder Formelmengen, deren Beziehungen zueinander, sowie von den jeweiligen Zuständen der Ableitungsketten. Ein logisches Zustandsübergangssysteme besteht aus einer Menge S von Zuständen und einer binären Relation $\rightarrow$, der Übergangsrelation. Jeder Zustand ist dabei die Repräsentation einer Formelmenge, im einfachsten Fall lediglich die Menge selbst. In ausgefeilteren Systemen enthalten die Zustände aber noch mehr Komponenten, Information über die Geschichte der Ableitung, Repräsentationen der von dem aktuellen Zustand aus machbaren Ableitungen zusammen mit strategischer und heuristischer Information über den Nutzen des jeweiligen Schritts usw. Die Übergangsrelation $S \rightarrow S'$ für Zustände S und S' ergibt sich aus dem Kalkül zunächst mal ganz einfach folgendermaßen: Wenn $\frac{F_1, \ldots, F_n}{F}$ eine Kalkülregel ist und $F_1, \ldots, F_n$ im Zustand S repräsentiert ist dann ist S' eine Repräsentation für $\mathcal{F} \cup \{F\}$, wobei $\mathcal{F}$ die in S repräsentierte Formelmenge ist. Auf dieser Ebene werden oft zusätzliche Operationen eingeführt, etwa das Löschen von redundanten Aussagen, so daß im Kalkül noch mögliche Ableitungen nun nicht mehr möglich – und hoffentlich auch nicht mehr nötig – sind. Die Übergangsrelation $\rightarrow$ enthält daher nicht nur Ableitungsschritte, sondern auch Reduktionsschritte, die die Formelmenge von unnützem Ballast befreien sollen.

Für jedes logische Zustandsübergangssystem sind drei Eigenschaften von Interesse: Korrektheit, Vollständigkeit und Konfluenz. Durch neue Regeln, mit denen Formeln oder Teile davon gelöscht werden, wird die Korrektheit und Vollständigkeit des zugrundeliegenden Kalküls beeinflußt. Für jede solche Regel muß man neu beweisen, daß man damit keine falschen Beweise erzeugt (Korrektheit), und daß man mögliche Beweise auch finden kann (Vollständigkeit). Die letzte Eigenschaften der Konfluenz ist für die Implementierung eines Deduktionssystems wichtig. Konfluenz bedeutet, daß wenn man von einem Zustand S aus zwei Nachfolgezustände S_1 und S_2 erreichen kann, dann gibt es für S_1 und S_2 einen gemeinsamen Nachfolgezustand S'. Ist ein System konfluent, dann kann man es sich bei der Suche leisten, zunächst mal in die falsche Richtung zu suchen. Wenn vom Ausgangszustand (S) überhaupt ein Weg zu einem Beweis (S') existiert, dann existiert er auch von jedem weiteren Zustand (S_1 oder S_2) aus. Das heißt, bei der Suche gibt es keine Sackgassen; man braucht nie mehr zu früheren Zuständen zurückzukehren (kein Backtracking). Das entspricht auch der Intuition für Beweissuchen: Eine Aussage, die man einmal abgeleitet hat, mag zwar nutzlos für den aktuellen Beweis sein, es sollte aber nicht nötig sein, deren Ableitung selbst ungeschehen zu machen. Vollständigkeit des Deduktionsprozesses bedeutet, daß man unabhängig von den ausgeführten Transformationen vom aktuellen Zustand aus den Beweis noch finden kann. Hat man sehr starke Reduktions- oder Steuerungsregeln, so kann das Zustandsübergangssystem des Deduktionssystems die Eigenschaft der Konfluenz verlieren. In diesem Fall bedeutet Vollständigkeit des Deduktionssystems, daß man nach der Ausführung gewisser Transformationen den Beweis nur noch dadurch findet, daß man zurücksetzt in einen vorangegangenen Zustand und eine andere Alternative wählt.

Steuerung Die letzte Schicht eines Deduktionssystems, die *Steuerung*, enthält schließlich die Strategien und Heuristiken, mit denen unter den möglichen Ableitungsschritten die jeweils sinnvollen ausgewählt werden. Hier steckt die eigentliche "Intelligenz" des Systems. Die Idee dabei ist, "gute" Schritte zu bevorzugen und "schlechte" Schritte zu vermeiden. Man unterscheidet Restriktionsstrategien und Ordnungsstrategien.

Restriktionsstrategien schränken den Suchraum weiter ein, in dem sie bestimmte Ableitungen einfach generell verbieten. Zum Beispiel verbietet die "Set of Support"-Strategie bei Widerlegungsbeweisen Ableitungen zwischen Axiomen untereinander. Die Idee dabei ist, daß bei Ableitungen zwischen Axiomen allein garantiert kein Widerspruch zu finden ist. Da Restriktionsstrategien bestimmte Ableitungen prinzipiell verbieten, muß man jeweils nachweisen, daß im verbleibenden Suchraum immer noch ein Beweis zu finden ist (Vollständigkeit).

Ordnungsstrategien sortieren die möglichen Ableitungen nach bestimmten Kriterien, z.B. könnte man Ableitungen, die kleine Formeln erzeugen, bevorzugen. Der Zweck der Ordnungsstrategien ist, die Suche selbst geschickt zu organisieren und natürlich die Suche zu terminieren, d.h. wenn es einen Beweis gibt, dessen Auffindung mit endlich vielen Schritten zu garantieren (Terminierung).

Verwendung, Dialogverhalten Ein wichtiger praktischer Aspekt ist die Art der Verwendung des Systems, bzw. die Art der Unterstützung die ein Deduktionssystem bietet:

Vollautomatisch Das Deduktionssystem hat als Eingabe die Axiome und die nach zuweisende Schlußfolgerungen, und als Ausgabe nur Erfolg/Mißerfolg und eine Begründung (Beweis). Dies ist die Idealvorstellung eines Automatischen Deduktionssystems. Pragmatische System beschränken sich auf eine (einfache) Auswahl) von Eingaben, z.B. einfache Verifikationsbedingungen. Im allgemeinen haben die Automatischen Deduktionssystems eine Menge von Parametern, die vor dem Lauf eingestellt werden müssen. So daß z.B. die Arbeit mit dem System Otter darin bestehen kann, in vielen Läufen, die richtigen Parameter bzw. Heuristiken einzustellen, so daß es am Ende doch eher interaktiv wirkt.

Es gibt Varianten dieser Systeme, die während der Suche interaktive Hilfestellung erlauben.

Halbautomatisch Bestimmte Systeme für Logiken höherer Ordnung (Typenlogik) führen einen Dialog mit dem Benutzer. Einfache Folgerungen werden automatisch durchgeführt, für andere wird von Benutzer erwartet, die Zwischenziele (Lemmas) zu formulieren und Beweistaktiken zu programmieren. Der geführte Beweis ist im Erfolgsfall korrekt, allerdings kann ein solcher Dialog einige Wochen dauern. Z.B. ist mit dieser Methode der Nachweis der Korrektheit eines Tokenizers von T. Nipkow (TU München) durchgeführt worden

Beweisprüfung Diesen Systemen muß die komplette Axiomatisierung und der Beweis vorgelegt werden. Dies kann selbst für einfache Dinge sehr mühsam sein, da der Detaillierungsgrad sehr hoch gewählt ist: (Jeder (formale) Schritt des Beweises muß formal korrekt vorliegen).