Aus: W.U. Dressler, W. Meid (Hrsg.): *Proceedings of the Twelfth International Congress of Linguists*. Vienna, August 28 - September 2, 1977. Innsbruck 1978, S. 589 - 591.

PETER HELLWIG

Logisch-funktionale Sätze natürlicher Sprache im Rahmen eines automatischen Deduktions-Systems

1. Im Folgenden möchte ich einige Gedanken vortragen, die dem neuen Programmsystem PLAIN zugrunde liegen. PLAIN (**P**rogram for **L**anguage **A**nalysis and **IN**ference) besteht aus zwei Komponenten: einem Analyseteil und einem Deduktionsteil.

2. Die Analyseprogramme ordnen jedem wohlgeformten Ausdruck der natürlichsprachigen Eingabe einen oder mehrere Ausdrücke in einer Konstruktsprache zu[1]. Die Ausdrücke der Konstruktsprache sind Dependenzgraphen. Als solche genügen sie den üblichen Bedingungen für gerichtete Baumdiagramme. Außerdem gilt, daß alle Knoten entweder mit Symbolen für Lexeme oder mit Variablen etikettiert sind. Die Kanten der Graphen repräsentieren die syntaktische Struktur der Äußerungen. Variablen können an Stelle der Etiketten einzelner Knoten oder an Stelle untergeordneter Teilbäume gesetzt werden. Baumdiagramme, die Variablen enthalten, heißen „Muster". Ausdrücke, die dadurch entstehen, daß in einem Muster alle Variablen durch Lexeme oder Lexemkonfigurationen ersetzt werden, sind „Instanzen" des Musters. Da die Typeneinteilung der Variablen eine rein graphtheoretische ist, bedarf es besonderer Vorkehrungen, um für den Deduktionsteil eine funktional adäquate Zuordnung von Instanzen zu Mustern zu gewährleisten. Zu diesem Zweck sind alle Knoten, zusätzlich zu den Symbolen für Lexeme oder Variablen, mit einer Rollenmarkierung versehen. Rollen sind „logische Konstituenten", d. h. syntaktische Einheiten, die mit Hilfe von Kommutationsproben im Rahmen von Schlußfolgerungen ermittelt werden[2].

3. Die Deduktionskomponente von PLAIN hat die Simulation natürlichsprachigen Folgerns zum Ziel. Die Logik der Konstruktsprache soll daher die Logik der natürlichen Sprache so genau wie möglich abbilden. Das heißt z. B., daß Paradoxien. die in der Eingabesprache

[1] Eine Darstellung des Analyseverfahrens unter dem Titel *Dependentielle Sprachanalyse* ist in Vorbereitung.

[2] Vgl. dazu Hellwig (1977) 66ff.

auftreten, durch PLAIN nicht etwa vermieden, sondern vielmehr gerade offenbar gemacht werden. Eine „Datenbasis" (D) ist im Modell von PLAIN eine Menge von Sätzen, die für einen bestimmten Gegenstandsbereich für wahr gehalten werden. Da nur deduktive Operationen simuliert werden sollen, bleibt die Referenz der Sätze selbst außerhalb des Modells. Ein Deduktionsproblem läßt sich nun generell definieren:

(i) Zu einem neu eingelesenen Satz p sind alle daraus in D folgenden Sätze k_1 bis k_n zu erzeugen, zusammen mit dem Ableitungsweg.

(ii) Zu einem neu eingelesenen Satz k sind alle Sätze p_1 bis p_m zu erzeugen, aus denen k in D folgt, zusammen mit dem Ableitungsweg.

Das erste Problem ist das der Schlußfolgerung, das zweite das der Beweisführung.

4. Ein vielbenutztes Verfahren zur Lösung von Deduktionsproblemen ist das sogenannte *theorem proving*. Der Grundgedanke dabei ist, daß die Verknüpfung der Prämissen und der Konklusion eines gültigen Schlusses mittels Implikation stets eine Tautologie ergibt. Um die Gültigkeit eines Schlusses festzustellen, berechnet man, ob die Verknüpfung der entsprechenden Sätze ein logisch-wahrer Ausdruck ist. Schwierigkeiten dieses Ansatzes liegen darin, daß im Falle des Deduktionsproblems (i) die ja nicht vorgegebenen Konklusionen irgendwie erzeugt werden müssen und daß im Falle (ii) möglicherweise die Sätze der gesamten Datenbasis mit dem in Frage stehenden Satz zu verknüpfen sind. Hinzu kommt, daß die Berechnung des Wahrheitswertes nicht abbricht, wenn eine Verknüpfung in Wirklichkeit kein Theorem ist[3].

Eine grundsätzliche Alternative besteht darin, für den Übergang von Prämissen zu Konklusionen syntaktische Regeln aufzustellen. Die vertrautesten Regeln dieser Art sind die sogenannten Schlußfiguren wie z. B. der *modus ponens*:

(1) *wenn* α *dann* β.
α.

β.

(1) ist eine metalogische Aussage und läßt sich lesen wie folgt: „Ist ein Satz der Form ‚*wenn* α *dann* β.' gegeben und ist ein Satz der Form ‚α.' gegeben, so kann ein Satz der Form ‚β.' erzeugt werden." Eine Schlußfigur ist also eine Transformationsregel mit zwei Satzbeschreibungen und einer Konstruktionsvorschrift. Alle logischen Gesetze, die Äquivalenzen oder Implikationen sind, können in derartige Transformationsregeln überführt werden[4].

5. Es wäre aber noch zu aufwendig, wollte man in D auf willkürliche Weise die Paare von Sätzen suchen, die mit dem Beschreibungsteil einer Schlußfigur übereinstimmen. Intuitiv ist klar, daß die Satzmuster Gemeinsamkeiten aufweisen, die für den Suchprozeß fruchtbar gemacht werden können. Unter diesem Gesichtspunkt vielversprechend erscheint die Wiederaufnahme der seit der Antike bekannten Unterscheidung von Minor-Prämisse und Major-Prämisse in einem Schluß. Die Minor-Prämisse macht eine Aussage über einen bestimmten Fall. Die Major-Prämisse dagegen ist eine Aussage über einen Zusammenhang, zu dessen Vorbereich die Minor-Prämisse und zu dessen Nachbereich die Konklusion des Schlusses eine Instanz bilden. Es ist das Gegebensein dieses Zusammenhanges, das dem Folgernden die Berechtigung gibt, von der Minor-Prämisse zur Konklusion überzugehen. S. E. Toulmin nennt die Major-Prämisse daher treffend *warrant* und *inference licence*[5].

---

[3] Vgl. Robinson (1971) 4.

[4] Zu Einzelheiten vgl. Bocheński/Menne (1965) 50ff.

[5] Toulmin (1958) 98. Ich teile nicht die Auffassung Toulmins, daß die Einteilung in Minor-Prämissen und Major-Prämissen "misleadingly few in number" sei. A. a. O. 96. Die übrigen Faktoren, die Toulmin aufzählt, können als Bestandteile einer komplexen Major-Prämisse berücksichtigt werden.

Es ist ein kleiner Schritt, um von Major-Prämissen zu Folgerungsalgorithmen zu kommen. Es bedarf dazu jeweils einer Metaregel, die z. B. für die Major-Prämisse des *modus ponens* so aussieht:

(2) ((wenn α dann β.)$\rightarrow$(α. $\vdash$ β.))

‚$\rightarrow$' ist zu lesen als „ersetze", ‚$\vdash$' ist zu lesen als „leite logisch ab". (2) ist eine Transformationsregel, deren Beschreibungsteil aus einem Satzmuster und deren Erzeugungsteil aus einem Muster für eine Folgerungsregel besteht. Jedes Mal, wenn ein eingelesener Satz eine Instanz zum Beschreibungsteil von (2) ist, kann eine Instanz der Folgerungsregel ‚(α.$\vdash$ β.)' konstruiert werden. In PLAIN speichern wir diese aktuellen Folgerungsregeln ebenfalls in der Datenbasis. Letztere enthält also neben Sätzen auch konkrete Folgerungsanweisungen. Tritt nun ein Satz p auf, der eine Instanz zum Beschreibungsteil einer Regel des Typs ‚(α. $\vdash$ β.)' ist, so dient er als Minor-Prämisse, und es kann eine Instanz von ‚β.' als Konklusion abgeleitet werden. Indem abgeleitete Sätze im nächsten Schritt wieder als Instanzen mit dem Beschreibungsteil weiterer Folgerungsregeln in D verglichen werden, läßt sich das Deduktionsproblem (i) zielstrebig lösen. Zur Lösung des Problems (ii) werden die in D gespeicherten Folgerungsregeln in umgekehrter Richtung angewandt. Zuerst werden die Regeln gesucht, deren Nachbereich ein Muster von k ist. Zu jeder anwendbaren Regel wird eine entsprechende Instanz des Vorbereichs erzeugt. Das Verfahren iterativ fortsetzend, erhält man schließlich alle Sätze, aus denen k nach den *warrants* in D überhaupt folgen kann. Am Ende wird nachgesehen, ob einer dieser Sätze tatsächlich in der Datenbasis steht. Das Programm ist effektiv, da es vom zu beweisenden Satz ausgeht und daher nur Sätze zusammenstellt, die wirklich zum Beweisziel führen können.

6. Der wesentliche Vorteil des geschilderten Deduktionsverfahrens ist es, daß keine Einschränkungen hinsichtlich der Syntax der verwendeten Sprache existieren. Daher eignet es sich besonders für natürliche Sprachen. Sätze einer natürlichen Sprache, die in Schlußfolgerungen als Major-Prämissen dienen können, nenne ich „logisch-funktionale" Sätze. Sie zeichnen sich dadurch aus, daß man sie mit Hilfe von Kommutationsproben auf Satzmuster zurückführen kann, für die sich Regeln zur Ableitung von Folgerungsregeln, wie (2), formulieren lassen. Die Konstanten in diesen Mustern sind im Deutschen Elemente wie *alle, jeder, kein, manche, wenn — dann, falls, wer — der, ist, nicht* und viele andere. Durch die Aufstellung von Regeln wie (2) wird die logische Syntax dieser Elemente sowie die der Strukturen, in denen sie vorkommen, beschrieben. Die Konstruktsprache von PLAIN ist ihrerseits flexibel genug, um die syntaktisch wie semantisch vielfältigen Arten logisch-funktionaler Sätze abzubilden.

Peter Hellwig, Kastellweg 21, D-69 Heidelberg

Bibliographie

Bocheński, I. M./Menne, A. (1965): Grundriß der Logistik. Paderborn (3. Aufl. 1965).

Hellwig, P. (1977): „Ein Computermodell für das Folgern in natürlicher Sprache". In: P. Eisenberg (Hrsg.): Semantik und künstliche Intelligenz. Beiträge zur automatischen Sprachbearbeitung II. Berlin, New York. 59–85.

Robinson, J. A. (1971): "Building Deduction Machines". In: N. V. Findler/B. Meltzer (Hrsg.): Artifical Intelligence and Heuristic Programming. Edinburg. 3–13.

Toulmin, S. E. (1958): The Uses of Argument. Cambridge.