Martina MARATSCHNIGER

KONTROLLRELATIONEN UND IHRE IMPLEMENTIERUNG IN TURBO-PROLOG. EINE COMPUTERLINGUISTISCHE STUDIE

1. EINLEITUNG

Dieser Artikel wird eine logisch-deklarative Algorithmisierung von Kontrollstrukturen im Rahmen der NTS (Natürlichkeitstheoretische Syntax)[1] vorstellen, die uns mit Fragen semantischer und struktureller Natur konfrontiert, welche sich nicht nur auf die zugrundeliegende Syntaxtheorie, sondern (v.a.) auch auf semanto-syntaktische Gegebenheiten/Möglich-keiten der verwendeten Programmiersprache (der KI) - TURBO-Prolog - beziehen.

2. KONTROLLRELATIONEN ALLGEMEIN

Sie werden in der *Kontrolltheorie* ((KT) = Teiltheorie der Rektions- und Bindungstheorie) behandelt, die die strukturellen Bedingungen für die Interpretation des *logischen Subjekts* (PRO) von Infinitivgruppen beschreibt. Koreferenzbeziehungen zwischen PRO und dessen Antezedenten werden durch einen spezifischen Interpretationsmechanismus zwischen diesen beiden Elementen geregelt. Chomsky (1981) unterscheidet zwischen *Objekts- und Subjektskontrolle*. Ruzicka (1982) bezieht das System der *thematischen Rollen* in die Diskussion um die Kontrollrelationsbestimmungen mit ein. Im Folgenden wollen wir die einzelnen Kontrollrelationen NTS-axiomatisch/TURBO-Prolog-algorithmisch betrachten.

2.1. Subjektskontrolle (SK)

Bei dieser Kontrollrelation sind Matrixsatzsubjekt und PRO koreferent. Ein typisches Subjektskontrollverb (SK-V) ist z.B. *versprechen*. Betrachten wir uns dieses Verb nun etwas genauer und gliedern NTS-gemäß zweistufig: *thematisch* und *kategorial*.

kategoriale Charakterisierung (KC) erfaßt Elemente innerhalb der Subkategorisierungsrahmen
thematische Charakterisierung (TC) zeigt die vom funktionalen Element selegierten θ-Rollen

Abb. 1: 2-Stufengliederung der Lexeme - NTS-axiomatisch

[1] Zum ersten/einzigen Implementierungsansatz der NTS vgl. Maratschniger (1992), bzgl. der NTS vgl. Mayerthaler/Fliedl (1989).

versprech_[V°] *„ich verspreche dir, mich zu schonen“*

LE: versprech- $[V^0]$

KC: $\left\lceil (NP_1) \left\{ \begin{matrix} NP_2 \\ S^1\ [\alpha\ FIN]_2 \end{matrix} \right\} \right\rceil$

TC: $\langle AG, Ziel_1, TH_2 \rangle$

Abb. 2: *versprechen* als Subjektskontrollverb

versprech_[AUX] *„das Konzert verspricht gut zu werden“*

LE: versprech- $[V^0]$

KC: [$(zu\ V^0)_{INFINITIV}$-]

TC: [———]

Abb. 3: *versprechen* als Auxiliar (= [+markiertes] Pendant zum [-markierten] SK-V)

Nun zu einem SK-NTS-P-Marker (vgl. Abb.4), den wir in das Modul SUBKON1.PRO implementieren, und zwar mittels der folgenden TURBO-Prolog Algorithmen (*clauses*):

```
v3(v3(V2),L1,L2):-v2a(V2,L1,L2).
v3sk(v3(V2),L1,L2):-v2b_sk(V2,L1,L2).
v2a(v2a(N3,V1),L1,L3):-n3a(N3,L1,L2),v1a(V1,L2,L3).
v2b_sk(v2b(N3,V0),L1,L2:-n3b_sk(N3),v0_inf(VO,L1,L2).
n3a(n3a(N0),L1,L2): -n0_i(N0,L1,L2),-n0_j(N0,L1,L2).
n3b_sk(n3b(PRO)):-n0sk(PRO).
n0sk(n0("PRO_i")).
v1a(v1a(V0,V1),L1,L3):-v0sk(V0,L1,L2),v1b(V1,L2,L3).
v1b(v1b(N3,V3),L1,L3):-n3a(N3,L1,L2),v3sk(V3,L2,L3).
n0_i(n0_i(No),[N0|L],L):-N0=er.
n0_j(n0_j(N0),[N0|L],L):-N0=ihm.
v0_inf(v0_inf(V0),[V0|L],L):-V0=zu_kommen.
v0sk(v0(V0, [V0|L],L):-V0=verspricht.
```

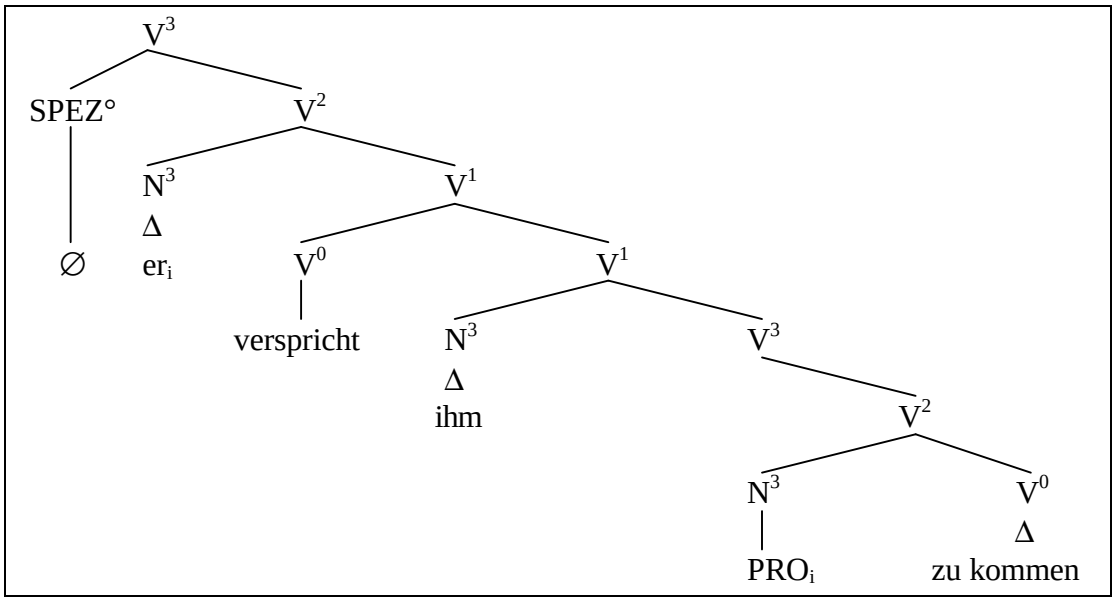

Abb. 4: SUBJEKTSKONTROLLE - *thematische Identität* (Implementierungsvorlage)

Welche Implementierungskonzepte mußten beachtet werden?

1. Es liegen zwei V^3 vor (→ clauses-mäßig: v3/v3sk):

- Die erste (= Träger-V^3) bildet den Ausgangspunkt der Konstruktion, trägt daher in der Programmimplementierung (= V3) keinen Index und ist somit in den domains definiert.
- Die zweite V^3 (= eingebettete (Sub)Struktur) ist strukturell/intern gleich aufgebaut wie die Träger-V^3, d.h. sie verzweigt in V^2. Somit wird sie erst in den *predicates* notiert, wo sie den Index sk (= Subjektskontrolle) erhält.

2. Es liegen zwei V^2-Varianten vor, die von den jeweiligen [± indizierten] V3 ausgehen; ihre Implementierung ist relativ komplex, da:

I. das Programm zwei (positionell) verschiedene (komplementär verteillte) V^2 erkennen muß;

II. das Programm den internen Unterschied der beiden V^2 repräsentieren soll/muß:
   - die eine spaltet sich nach rechts in N^3 auf, das in ein N^0 übergeht, welches ein lexikalisiertes Element dominiert und nach links in ein V^1 verzweigt;
   - die andere V^2 geht nach links ebenfalls in N^3 über, dieses ist aber mit dem zuvor beschriebenen N^3 nicht identisch, da es in das, von einem N° dominierte, nicht-lexikali-sierte/terminale Element (= logische Subjekt, leere Element) PRO übergeht. Nach rechts verzweigt die zweite V^2 bedingt durch pruning in V°.

   Diesem Unterschied wird man algorithmisch gerecht, indem man die Varianten v2a/v2b bereits in den clauses spezifiziert, jener Ebene, die nicht nur (I) sondern auch die Konzeption unter (II) formalisiert: v3 geht über in v2a, v3sk in v2b. Um anzuzeigen, daß v2b PRO indirekt dominiert (= mit Subjekt der ersten V^3 koreferent) fügen wir dieser V^2 (durch einen underline getrennt) den Index sk hinzu. Nun gilt es noch

festzulegen, welcher Index in der Ausgabestruktur (Userebene = programmextern) [α erscheint] (wobei α=+/-):[2] sk = programminterne Gliederung → kein screen-output
b = [α programminterne Gliederung] → screen-output
Damit haben wir die V^2-Ebenen-Implementierung abgeschlossen.

3. Die V^1-Rekursionsimplementierung regelt man durch Variantenschreibung (= v1a/v1b).
4. Wie in (II) bereits erwähnt, liegen in der Struktur zwei N^3 Varianten vor:

- Die erste - n3a - geht in ein lexikalisches Element über (= terminale Ebene), das indiziert werden muß (vgl. Punkt 5).
- Die zweite - n3b - geht über in ein N°, welches PRO (das wiederum zu indizieren ist, vgl Punkt 5) direkt dominiert. n3b muß konsequenterweise zusätzlich mit dem Index sk spezifiziert werden. Damit die Ausgabestruktur n3b enthält, setzen wir diese clauses-Zeilen an: n3b_sk(n3b(PRO)):-n0sk(PRO).
  Damit können wir den nächsten Schritt abarbeiten.

5. n0 + PRO sind [α programmintern] aufeinander zu beziehen → 4 Varianten für N°:[3]

- n0_i zur Kennzeichnung des lexikalischen Elements in Subjektsposition;
- n0_j zur Identifizierung des lexikalischen Elements in Objektsposition;
- n0sk zur Einbindung von PRO in die Subjektskontrollrelation;
- n0 um PRO von N° in der Ausgabestruktur dominieren zu lassen.

  PRO (terminales Element) wird mittels *i* indiziert (→ ein-eindeutige Anzeige der SK). n0_i/n0_j müssen in die gleiche lexikalische Ebene übergeführt werden, und zwar so, daß die Indizierung auch in der Ausgabestruktur aufscheint; erst dann wird die Parsingprozedur (automatische Klammergenerierung) der/einer Subjektskontrolle (später dann im Vergleich zur automatischen Indexzuweisung und Klammererstellung im Rahmen aller anderen Kontrollrelationen) relevant und NTS-systemadäquat.

6. Schließlich sind noch 3 Verbvarianten einer Algorithmisierung zu unterziehen:

- v0_sk zur eine-eindeutigen Identifizierung des Subjektskontrollverbs;[4]
- v0_inf um den Verbteil in der Domäne von v3sk spezifizieren zu können;
- v0 damit in der Ausgabestruktur das terminale Element (= *versprechen*) von v0 und nicht von v0sk dominiert wird.

  Vgl. hierzu die nachstehenden beiden clauses-Zeilen miteinander:
  v0_inf(v0_inf(V0),[V0|L],L):-V0=zu_kommen.

---

[2] Diese Überlegungen ergaben dann die clauses-Zeile für v2b_sk.

[3] Vgl. zur Implementierung von PRO Maratschniger/Fliedl (1993a).

[4] Später muß für alle Kontrollrelationen auch die Tatsache implementiert werden, daß die Existenz eines Verbalkomplexes (Verbindung: mindestens ein finites mit mindestens einem infiniten Verb, durch Statusrektion miteinander verbunden) bedeutungsunterscheidend/-verändernd wirken kann (⇒ u.U. andere Struktur, weil u.U. andere Kontrollbeziehung).

```
v_sk(v0(V0),(V0|L],L):-V0=verspricht.
```

Somit ist das Problem Subjektskontrolle regelgerecht und sauber implementiert. Oder? Wie sieht es aber nun mit einem Satz wie *Er verspricht ihm zu helfen* aus? Zunächst scheint es, als ob die entsprechende Klammerstruktur hierzu mittels des soeben erstellten Parserteils und einer entsprechenden Lexikonerweiterung problemlos generiert werden könnte. Ist dies aber wirklich so? Nein! Denn bei genauerer Betrachtung fallen 6 Probleme auf:

1. Wir haben in dem gerade angeführten Beispiel kein Komma gesetzt. →
2. Man kann nicht genau erklären ob *ihm* Objekt von *versprechen* oder von *helfen* ist. →
3. Es sind zwei Strukturen in einem Satz. Bei V° Tausch (*verbieten* statt *versprechen).* →
4. Es liegt keine Subjektskontrolle (SK) sondern Objektskontrolle (OK) vor. →
5. Diese beiden Strukturen (SK/OK) unterscheiden sich nur durch den Index bei PRO. ⇒
6. Strukturen können zwar am Papier durch Sonderzeichen (Kommas/Indizes) ein-ein-deutig gemacht werden, für den Computer bleiben sie aber dennoch mehrdeutig: er identifiziert (bzgl. 5.) die beiden Strukturen als identisch aufgebaut. → Ob PRO - ein terminales Element - nun den Index i oder j aufweist, ist eine rein semantische, nicht aber strukturelle Problematik (bzgl. 4.-6. vgl Kap. 2.2.).

Die Implemetierung integriert diese Problematik nicht, da man davon ausgehen kann, daß der User über kein Hintergrundwissen (hier: Interpunktion bei satzwert gen/inko-härenten Infinitivgruppe) verfügt.

Die zwei Objektsinterpretationen (→ Strukturen) zeigen die folgenden vier Struktu

Er verspricht ihm, zu helfen. = Er verspricht ihm$_j$ $_i$ zu helfen
ihm Objekt von)

Abb. 5

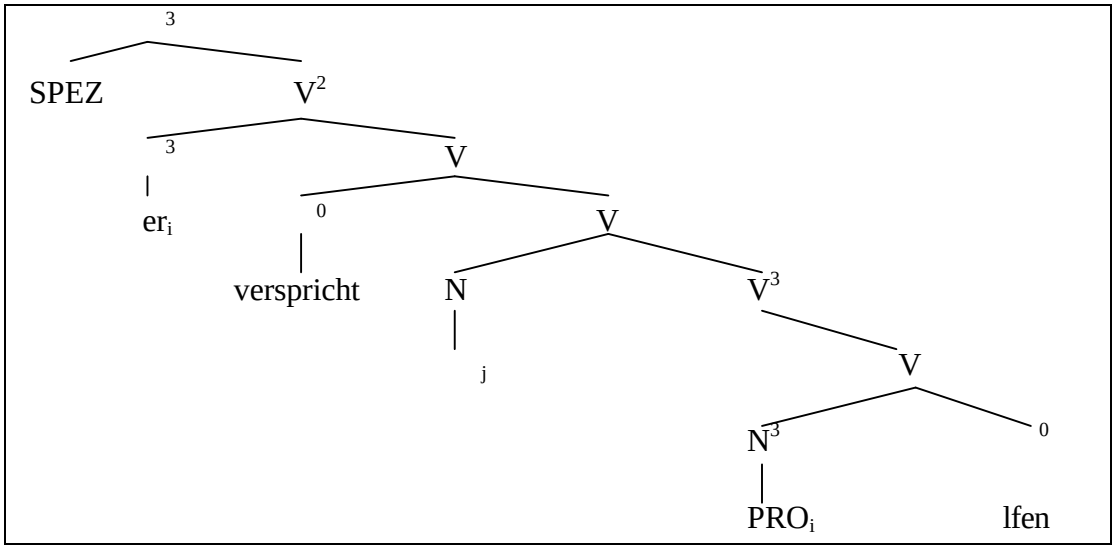

Abb. 6

Er verspricht ihm, zu helfen. = Er_i verspricht, PRO_i ihm_j zu helfen

(SK - thematische Identität, *ihm* Objekt von *helfen*)

Abb. 7

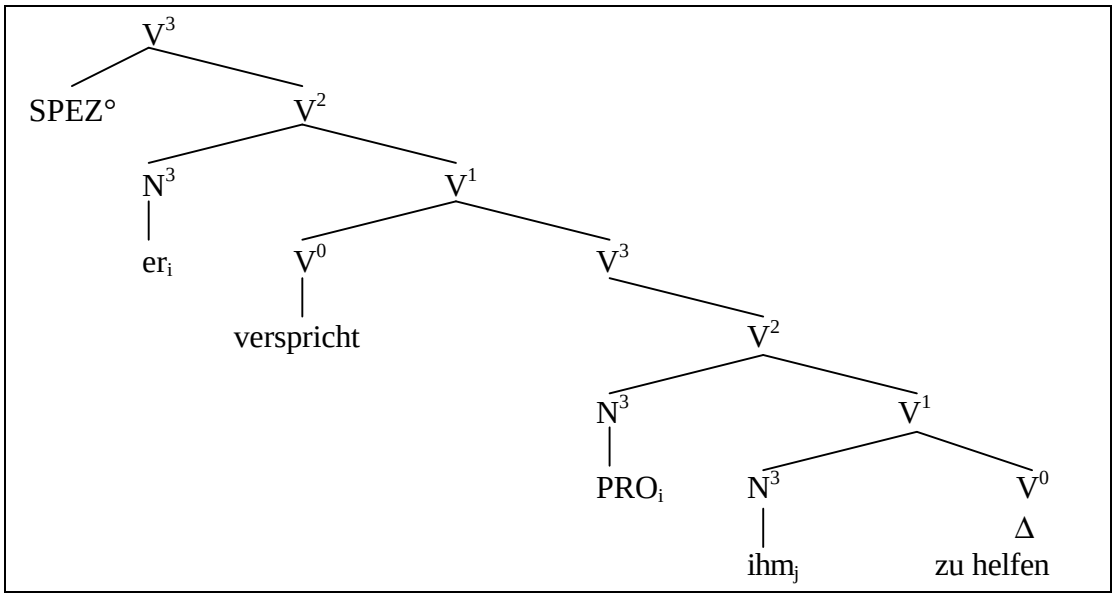

Abb. 8

ad 3. Den zuvor bereits diskutierten clauses-Teil für die SK muß man nun um die entsprechenden Implementierungsalgorithmen der zweiten Interpretationsvariante erweitern, bzw. diese in ein weiteres (zusätzliches) Modul (SUBKON2.PRO) integrieren:[5]

```
v3sk(v3(V2),L1,L2):-v2b_sk(V2,L1,L2);
v2a(v2a(N3,V1),L1,L3):-n3a(N3,L1,L2),v1c(V1,L2,L3);
v2a_sk(v2a(N3,V1),L1,L2):n3b_sk(N3),v1d(V1,L1,L2).
vlc(vlc(V0,V3),L1,L3):-v0sk(V0,L1,L2),v3sk(V3,L2,L3).
v1d(v1d(N3,V0),L1,L3):-n3a(N3,L1,L2),v0_inf(Vo,L2,L3).
v0_inf(v0_inf(V0),[V0|L],L):-v0=zu_helfen.
```

[5] SUBKON1.PRO, SUBKON2.PRO, wie auch alle folgenden Kontrollstrukturmodule, werden von einem Supermodul (KONTROLL.PRO) [α-direkt] dominert. SUBKON1/2 indirekt (alle anderen direkt), da diese beiden Subjektskontrollmodule durch ein eigenes, für sie erstelltes SUPERMODUL (SUBKON.PRO) direkt dominiert werden. SUBKON.PRO unterliegt der direkten Dominanz von KONTROLL.PRO.

Warum nun ein eigenes Modul für diese clauses? Würde man sie in SUBKON1.PRO integrieren und ließe man sich die Klammerstruktur(en) für *Er verspricht ihm zu helfen* und dann für *Er verspricht ihm zu kommen* ausgeben, erhielte man für den zweiten Satz ebenfalls zwei output-Varianten, die den Aufbauprinzipien des ersten Satzes gehorchen. Warum? Weil uns hier die Semantik in die Quere kommt.[6] Worin liegt nun ein Lösungsweg? Man muß dem Computer programmtechnisch vorinterpretieren wieviele und v.a. welche Kontrollrelationen er aus einem Satz lesen darf/kann/soll und welche nicht. Erst die Interpretation - die Erstellung einer "semantischen Struktur" (semantischen Form) - erlaubt die korrekte, automatisch generierte Konstruktion einer Klammerstruktur, denn der

"menschliche Parser"

weiß (zumeist "instinktiv"), daß: wenn *zu helfen* in der zweiten V^3 steht, es zwei Möglichkeiten gibt, den Satz zu verstehen. Um dies deutlich zu machen, setzt er Interpunktion und Intonation (Sprechpausen) ein; Ebenen, die beim Parserbau auf Computerebene (noch) nicht berücksichtigt werden (können/sollen). Im Falle von *zu kommen* in der zweiten V^3, kann *ihm* nur als Objekt von versprechen auftreten - nie als Objekt von *zu kommen*. Der

"maschinelle Parser"

muß daher zwei zunächst getrennte Module für die Subjektskontrolle (SUBJEKT1.PRO (= A) und SUBJEKT2:PRO (= B)) erhalten, die später programmintern zusammengeschlosssen werden (vgl hierzu Fußnote 5), und zwar so, daß der unversierte - nicht mit linguistischem Hintergrundwissen bestückte - User auf "nur" einer Programmebene alle Subjektskontrollstrukturen generieren kann, bzw. zur Verfügung hat:

(A) umfaßt Strukturen wie *Er verspricht ihm zu helfen.*
(B) umfaßt Strukturen wie *Er verspricht ihm zu kommen.*[7]

Das Implementierungsprinzip von/für Kontrollstrukturen (ausgehend von SK):
Die Modusaufspaltung erfolgt stets nach diesen (3) Kriterien:
1. Programmierung der Kontrollrelation des Matrixverbs;

---

6 Intonationsverläufe u.dgl. bleiben bei dieser Implementierung ausgeklammert.

7 Ein gewisser Redundanzgrad innerhalb von Regelkorpora ist notwendig/wünschenswert. Die zwei SK-Module sind dadurch untrennbar miteinander verbunden - ergänzen sich gegenseitig; was das eine Modul durch sein Inventar an Verben nicht haben kann/darf, hat das andere und umgekehrt. Einzelmodule sind verschieden informationsreich, isoliert voneinander vollkommen funktionsfähig; erst zusammen bilden sie ca. 100% der Gesamtinformation bzgl des SK-Potentials ab. Zusätzlich verweisen sie durch ähnliche/gleiche Algorithmen auf die anderen Kontrollrelationen-Module. So entsteht ein Netz, das schließlich den "gesamten" Bereich der Kontrollrelationen "in den Griff bekommt".

2. Zuordnung, modulspezifische Programmierung, Einordnung der (syntaktischen/semantischen) Einflüsse des "untergeordneten infinitivischen Verbalelements"[8] (in zweiter V^3) auf die Gesamtstruktur bzw. deren Interpretation (Kontrollrelationenänderung u.dgl.);
3. Zuordnung, modulspezifische Programmierung, Einordnung der (syntaktischen und semantischen) Einflüsse des "untergeordneten Verbalkomplexes" (in zweiter V3) auf die Gesamtstruktur bzw. deren Interpretation (Kontrollrelationenänderung u.dgl.).

2.2. Objektskontrolle (OK)

Ausgangspunkt hierfür war Kapitel 2.1.(6. Problem: 4). Der folgende Beispielsatz bildet die Implementierungsgrundlage: *Er verbietet ihm, zu kommen*. Nun zur kontrollrelationenspezifischen Abstraktion: Er_i *verbietet* ihm_j, PRO_j *zu kommen* und zum NTS-P-Marker:

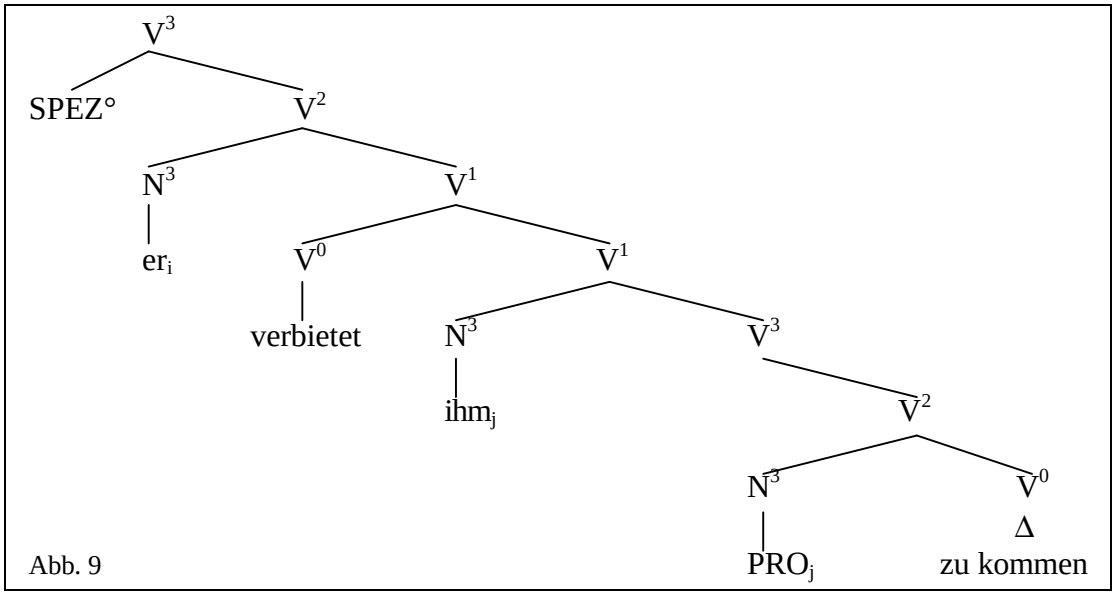

Abb. 9

Stellt man nun die P-Marker unter (4) und (9) gegenüber, werden die Probleme bzgl. 4. bis 6. in Kap. 2.1. deutlich. Der einzige Unterschied zur Subjektskontrolle im strukturellen Aufbau ist der Index von PRO. Das Objektskontrollmodul OBJEKT.PRO kann somit nach den programmtechnisch gleichen Algorithmen erstellt werden wie die Subjektskontrollmodule SUBKON1.PRO/SUBKON2.PRO, da die SK-Modul-Programmierung alle strukturellen Implementierungsüberlegungen für die Algorithmisierung

[8] Der Implementierungsvorgang bei Existenz von Verbalkomponenten in der zweiten V^3 wird hier nicht behandelt.

der restlichen Kontrollstrukturen mit sich gebracht hat. Für das OK-Modul sieht das dann clauses-mäßig so aus:

```
v3(v3(V2),L1,L2):-v2a(V2,L1,L2).
v3ok(v3(V2),L1,L2):-v2b_ok(V2,L1,L2).
v2a(v2a(N3,V1),L1,L3):-n3a(N3,L1,L2),v1a(V1,L2,L3).
v2b_ok(v2b(N3,V0),L1,L2):-n3b_ok(N3),v0_inf(V0,L1,L2).
n3a(n3a(N0),L1,L2):-n0_i(N0,L1,L2).
n3a(n3a(N0),L1,L2):-n0_j(N0,L1,L2).
n3b_ok(n3b(PRO)):-n0_ok(PRO).
n0_ok(n0("PRO_j")).
v1a(v1a(V0,V1),L1,L3):-v0_ok(V0,L1,L2),v1b(V1,L2,L3).
v1b(v1b(N3,V3),L1,L3):-n3a(N3,L1,L2),v3ok(V3,L2,L3).
n0_i(n0_i(N0),[N0|L],L):-N0=er.
n0_j(n0_j(N0),[N0|L],L):-N0=ihm.
v0_inf(v0_inf,(V0),[V0|L],L):-V0=zu_kommen.
v0_ok(v0(V0),[V0|L],L):-V0=verbietet.
```

Eine "Kumulation" aus Objektskontroll- und Subjektskontrollimplementierung ist die im nächsten Abschnitt behandelte *ambige Kontrolle*.

2.3. Ambige Kontrolle (AK)

Ein Satz wie *Er überzeugte ihn, fleissig zu sein* muß strukturell so interpretiert werden:

$$Er_i \text{ überzeugte } ihn_j, PRO_i \text{ fleissig zu sein.}$$

Abb. 10: Subjektskontrolle ⇒ thematische Identität

$$Er_i \text{ überzeugte } ihn_j, PRO_j \text{ fleissig zu sein.}$$

Abb. 11: Objektskontrolle ⇒ thematische Distinktheit

d.h., wir müssen folgende Gesamtinterpretation ansetzen:

$$Er_i \text{ überzeugte } ihn_j, PRO_{i,j} \text{ fleissig zu sein.}$$

Abb. 12: Ambige Kontrolle ⇒ thematische Identität/Distinktheit

Was dies nun für die strukturelle Repräsentation und demnach für die Implementierung bedeutet, zeigen die auf dem Strukturbaum der folgenden Seite basierenden clauses.

```
v3(v3(V2),L1,L2):-v2a(V2,L1,L2).
v3ak(v3(V2),L1,L2):-v2_ak(V2,L1,L2).
v2a(v2a(N3,V1),L1,L3):-n3a(N3,L1,L2),v1a(V1,L2,L3).
v2_ak(v2a(N3,V1),L1,L2):-n3b_ak(N3),v1c(V1,L1,L2).
v1a(v1a(V0,V1),L1,L3):-v0_ak(V0,L1,L2),v1b(V1,L2,L3).
v1b(v1b(N3,V3),L1,L3):-n3a(N3,L1,L2),v3ak(V3,L2,L3).
v1c(v1c(A2,V0),L1,L3):-a2(V0,L1,L2),v0_inf(V0,L2,L3).
n3a(n3a(N0),L1,L2):-n0_i(N0,L1,L2).
n3a(n3a(N0),L1,L2):-n0_j(N0,L1,L2).
n3b_ak(n3b(PRO)):-n0_ak(PRO).
n0_ak(n0("PRO_i,j")).
a2(a2(A0),L1,L2):-a0(A0,L1,L2).
n0_i(n0_i(N0),[N0|L],L):-N0=er.
n0_j(n0_j(N0),[N0|L],L):-N0=ihn.
v0_inf(v0_inf(V0),[V0|L],L):-V0=zu_sein.
v0_ak(v0(V0),[V0|L],L):-V0="überzeugte".
a0(a0(A0),[A0|L],L):-A0=fleissig.
```

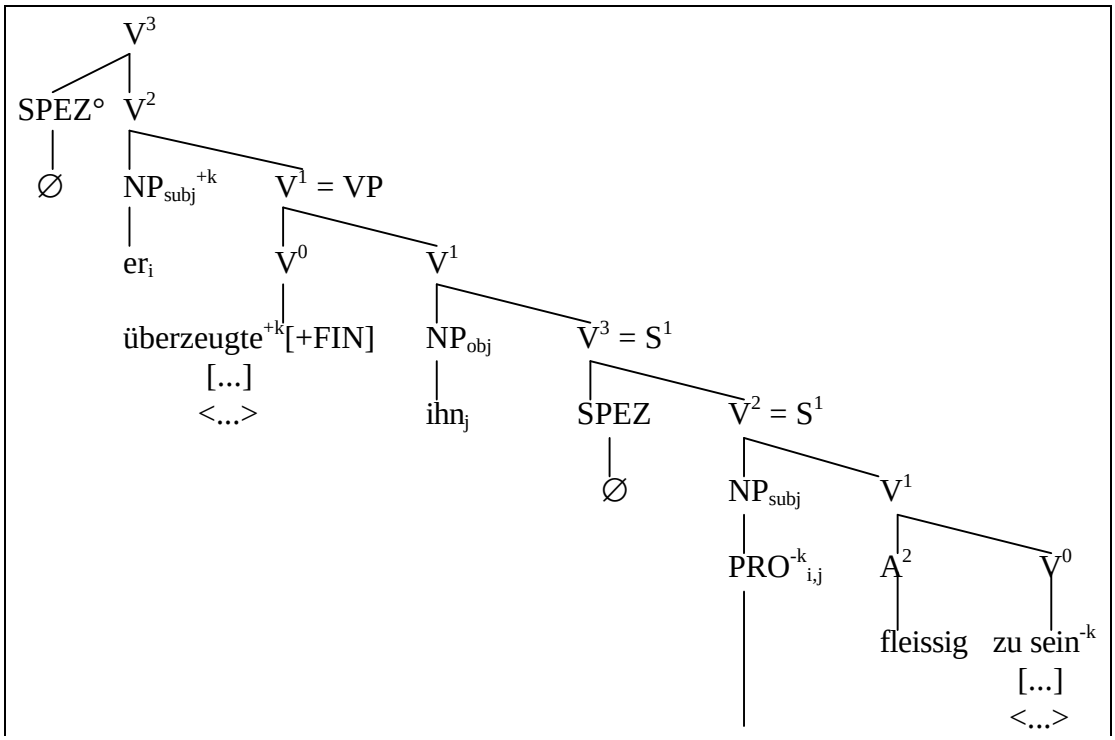

Abb. 13: Ambige Kontrolle - thematische Identität/thematische Distinktheit

2.4. Arbiträre Kontrolle (ARK)

Bevor wir einen Beispiel-P-Marker konstruieren, hier die Implementierungskonzepte:

1. Erstellen von 3v1-Varianten: v1a, v1b, v1c;
2. Anwendung einer v3-unterscheidenden Indizierung Xarb (arb erscheint nicht im output);
3. Index nur bei PRO, da keine Vorgängerkategorie;
4. Generierung des expletiven Subjekts.

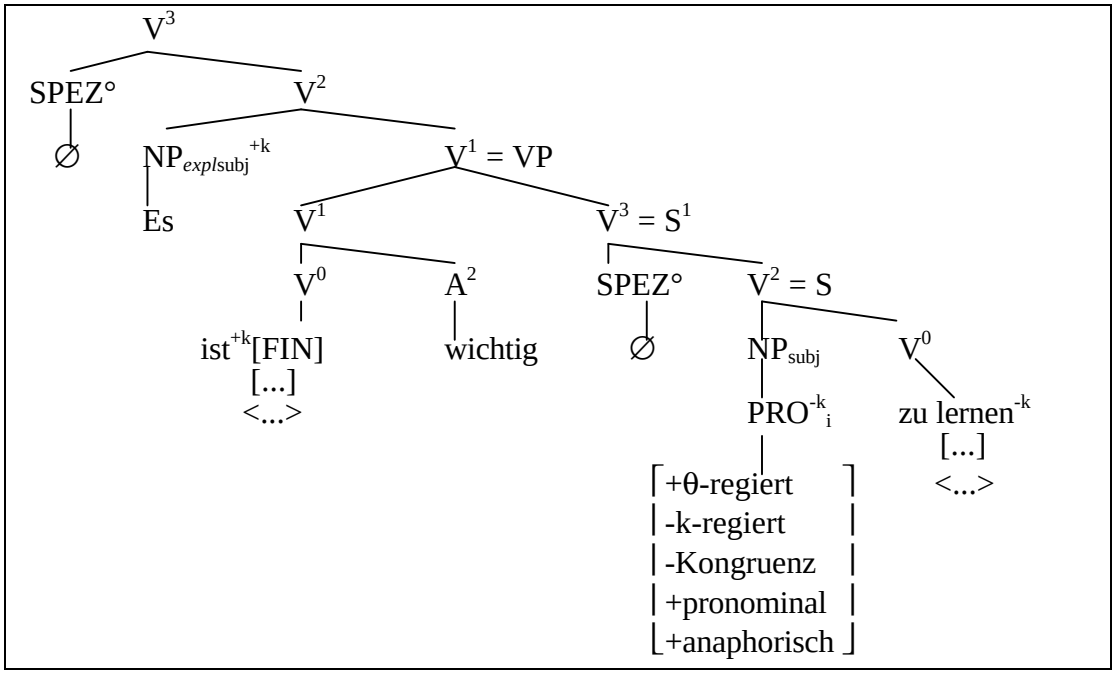

Abb. 14: Arbiträre Kontrolle: *Es ist wichtig, zu lernen.*

Hierzu nun die Haupt-clauses, die die Kontrollstrukturenimplementierung komplettieren:

```
v3ark(v3(V2),L1,L2):-v2ark(V2,L1,L2).
v2ark(v2a(N3,V1),L1,L2):-n3b_ark(N3),v1c(V1,L1,L2).
n3b_ark(n3b(PRO)):-n0ark(PRO).n0ark(n0("PRO_i")).
v1a(v1a(V1,V3),L1,L3):-v1b(V1,L1,L2),v3ark(V3,L2,L3).
```

```
v1b(v1b(V0,A2),L1,L3):-v0(V0,L1,L2),a2(A2,L2,L3).
v1c(v1c(V0),L1,L2):-v0=inf(V0,L1,L2).
a2(a2(A0),L1,L2):-a0(A0,L1,L2).
n0(n0(N0),[N0|L],L):-N0=es.
v0(v0(V0),[V0|L],L):-V0=ist.
v0inf(v0inf(V0),[V0|L],L):-V0=zu_lernen.
a0(a0(A0),[A0|L],L):-A0=wichtig.
```

3. ZUSAMMENFASSUNG

Im Rahmen der TURBO-Prolog-Implementierung von Kontrollstrukturen (KS) hat sich (bereits anhand der SK-Programmierung) gezeigt, daß es unmöglich ist, nur ein Programmodul für die Regelkorpora aller KS zu verwenden, da sich Semantik (semantische Interpretation) und Syntax (syntaktische Struktur) bzgl. der logisch deklarativen Algorithmisierung "in die Quere" kommen. Was am Papier ein-eindeutig erscheint, wird im Datenfeld zu einer ambigen Angelegenheit. Der KS-Implementierung muß eine Erfassung leerer Elemente vorangehen und eine Berücksichtigung semantischer Formen folgen. Erst so ist es möglich, linguistisches (Experten)Wissen aufzubereiten und dann sowohl einem versierten User, als auch einem Nicht-Linguisten als Arbeitsmaterial zur Verfügung zu stellen.

LITERATUR

Bratko, I.	1986	*Prolog Programming and Artificial Intelligence*, Addison Wesley
Chomsky, N.	1981	*Lectures on Government and Binding*, Dordrecht
Fliedl, G.	1988	Kontrollphänomene und thematische Rollen, in: *2. Jenaer Semantik-Syntax-Symposium* (= Wissenschaftliche Beiträge der Universität Jena)
Haider, G.	1984	Was zu haben ist und was zu sein hat: Bemerkungen zum Infinitiv, *PZL* 30
Jahn, M.	1989	Nichtnumerisches Programmieren mit ICON, in: Line, M.P./Wallmannsberger, J. (Hrsg.) *Innsbrucker Beiträge zur Kulturwissenschaft. Anglistische Reihe Bd. 3. Computer & Sprache: Papiere des Workshops Universität Saarbrücken, 23.-25. 11. 1989*, Innsbruck
Maratschniger, M./Fliedl, G.	1991	Natürlichkeitstheoretische Syntax - Ein Modell und seine Anwendung, in: *Betriebslinguistik und Linguistikbetrieb. Akten des 24. Linguistischen Kolloquiums Bremen 4.-6. September 1989. Band 1*, Tübingen
	1993	Zur Implementierung der "LEEREN" Subjekte "PRO" und "pro" in TURBO-Prolog im Rahmen der NTS, *PZL* 39/1
Maratschniger, M.	1992	*Zur Implementierung der NTS in TURBO-Prolog*, Klagenfurt (Diss.)
	1993	*Vom Satz zur NTS-Baumstruktur. Vom Wort zur NTMS-Baumstruktur. Eine computerlinguistische Studie anhand von TURBO-Prolog - PDC-Prolog - ICON - WordPerfect 5.1 (Projektbericht)*, Klagenfurt

Mayerthaler, W./Fliedl, G. 1989 Natürlichkeitstheoretische Syntax (NTS), in: Jacobs/v.Stechow/Sternefeld/Vennemann (Hrsg.*) Ein internationales Handbuch zeitgenössischer Forschung*, Berlin/New York

Ruzicka, R. 1982 Kontrollprinzipien infiniter Satzformen: Infinitiv und Gerundium im Russischen und in anderen slawischen Sprachen, *Zeitschrift für Slawistik* 27/3

Standke, R. 1987 *Turbo-Prolog. Abbild menschlicher Denkstrategien*, Hamburg

Martina MARATSCHNIGER
Institut für Sprachwissenschaft
Universität Klagenfurt