Martina MARATSCHNIGER

ZUR IMPLEMENTIERUNG DER NATÜRLICHKEITSTHEORETISCHEN SYNTAX (NTS) IN TURBO-PROLOG EINE COMPUTERLINGUISTISCHE STUDIE

0. EINLEITUNG

Im vorliegenden Aufsatz soll gezeigt werden, wie die grundlegenden Analysekonzepte der Natürlichkeitstheoretischen Syntax (NTS; siehe Mayerthaler/Fliedl 1993) in TURBO-Prolog übersetzt werden können.

Im ersten Teil werden kurz die grundlegendsten Merkmale der NTS skizziert sowie ein Einblick in die verwendete deklarative, logische Programmiersprache gegeben.

Im zweiten Teil geht es dann um die Beschreibung der, für die Übersetzung in TURBO-Prolog notwendigen, Programmierschritte sowie um grundsätzliche Fragestellungen der Implementierung.[1]

Durch die, dem Parser zugrundegelegten, Algorithmen ist es möglich, Klammerstrukturen zu beliebigen Sätzen (beliebiger Sprachen) vom Computer generieren zu lassen, wobei die Klammerung selbst eine exakte (genauer: "ein-eindeutige") Umsetzung der entsprechenden NTS-Baumstruktur ist und das gesamte Regelkorpus dieser Syntaxtheorie umfaßt - die graphische Umsetzung der Klammerstrukturen in P-Marker geschieht dann anhand eines Programmes aus der Programmiersprache ICON, mit einem dazwischengeschalteten WP51-Macro, durch welches die sprachspezifische Sonderzeichenproblematik (z.B. c, s, Â,..., α, usw.) gelöst wird (vgl. Maratschniger 1993c).

1.0. DIE NATÜRLICHKEITSTHEORETISCHE SYNTAX (NTS)

Sie ist eine generativistisch orientierte, verbzentrierte Syntaxtheorie, die die grammatische Realität als das Zusammenspiel von aufeinander beziehbaren Teilsystemen versteht. In der folgenden Modellstruktur kommt auch zum Ausdruck, daß die NTS in Analogie zur Rektions- und Bindungstheorie (vgl. Chomsky 1981)) entwickelt wurde.

---

[1] Der erste Ansatz zur Implementierung der *NTS* ("*Natürlichkeitstheoretische Syntax*") in *Turbo-PROLOG* wurde im Rahmen der Dissertation von Maratschniger (1992; Dissertation am Institut für Sprachwissenschaft der Universität Klagenfurt) vorgestellt. Es war dies der erste und auch einzige Versuch, die NTS mit Hilfe einer Programmiersprache zu erfassen.

LEXIKON
(Angereichert um eine natürlichkeitstheoretische Bewertungskomponente)

Morphosyntax TIEFENSTRUKTUR Satzsyntax
(modifizierte Version der X'-Theorie)

SEICHTSTRUKTUR
(Mit Spuren und Koindizierung angereicherte Tiefenstruktur)
(samt natürlichkeitstheoretischer Bewertung)

SEMANTISCHE FORM PHONOLOGISCHE FORM

Die tiefenstrukturelle Repräsentation wird mittels des NTS-spezifischen X'-Modells durchgeführt, welches u.a. dadurch charakterisiert ist, daß der Wert der zulässigen Maximalprojektionen (= X^{max}) von lexikalischen Elementen (N, V, ADJ, ...) *kategorienspezifisch* bestimmt werden muß. Wie dies im Einzelnen aussieht, zeigt die folgende Abbildung.

X^n ----------> ... X^n / X^{n-1}

wobei: X^n bei Kategorienrekursion
und X^{n-1} sonst

$0 \leq n \leq 3$ / [N,V]
$0 \leq n \leq 2$ / [A]
$0 \leq n \leq 1$ / [P]

n=0 / [SPEZ]

$V^{max} = V^3$
$N^{max} = N^3$
$A^{max} = A^2$
$P^{max} = P^1$

$SPEZ^{max} = SPEZ°$

Nominalphrasen (= N^3) und (spezifizierte) Sätze (= V^3, V^2 = unspezifizierter Satz bzw. Satzradikal) werden also als Maximalprojektionen von N bzw. V dargestellt. In Anlehnung an andere verbzentrierte Syntaxmodelle, wie z.B. die Valenzgrammatik, wird auch in der

NTS das *Verb* stets als Kopf des *spezifizierten* Satzes (= V^3) und das *Nomen* unmarkierterweise als Kopf der *Nominalphrase* (= N^3) gekennzeichnet. Nur echte lexikalische Kategorien wie N°, V°, A°, P° können als X°-Kategorien (= Keim oder Minimalprojektion = X^{min}) fungieren (sie sind kasusregierend). Lexikalische Kategorien werden im Rahmen der Natürlichkeitstheoretischen Syntax sowohl durch semantische als auch durch *morphosyntaktische* Merkmale subklassifiziert. Die Phrasenstrukturmarker, die dann auf diesen Merkmalsbündeln aufbauen, repräsentieren syntaktische Konstruktionen in der Form von *Entfaltungen* dieser Merkmalsbündel. Die erzeugten Phrasenstrukturen werden schließlich mit Hilfe der Prinzipien der Natürlichkeitstheorie (NT) *natürlichkeitstheoretisch* bewertet (vgl. Mayerthaler/Fliedl 1993) ein Vorgang, den es im Rahmen der TURBO-Prolog Implementierung der NTS ebenfalls algorithmisch zu erfassen gilt. Die Natürlichkeitstheoretische Syntax ist also ein Analysemodell, welches versucht, die Repräsentation der Konstituentenstruktur von Sätzen mit deren Natürlichkeitsgrad zu korrelieren. Der Grad der Komplexität von P-Markern (und Merkmalskonfigurationen) steht in einer umgekehrten Wechselbeziehung zum Grad der Natürlichkeit, d.h. je komplexer ein Satz ist, desto unnatürlicher ist er und umgekehrt. Die NTS ist somit bestrebt, auch jene Merkmale ausfindig zu machen, die konstruktionelle Unterschiede bewirken, da erst die Repräsentation dieser, eine natürlichkeitstheoretische Bewertung ermöglichen. Nur *konstruktionsimmanente* Merkmale bewirken die konstruktionelle [α-Natürlichkeit] eines Satzes (wobei: α = +/-).

2.0. WAS IST TURBO-PROLOG?

TURBO-Prolog ist eine prädikative Programmiersprache. Ihre Syntax erlaubt es, dem Computer ein bestimmtes Wissensfeld (= Wissensbasis) "vorzuformulieren" (z.B. den Regelapparat einer Syntaxtheorie), aus dem er dann die einzig mögliche Lösung/alle möglichen Lösungen in Bezug auf eine bestimmte Fragestellung (z.B. die syntaktische Struktur eines Satzes) ermitteln kann. Mit anderen Worten: In der Sprache der *formalen Logik* und somit innerhalb von PROLOG definiert man eine Wissensbasis als *endliche Menge von wahren Aussagen*.

Bis heute liegt keine einheitliche Definition der Sprache PROLOG vor und d.h. man muß sich dialektspezifisch mit den einzelnen Systemen auseinandersetzen, *Fakten* und *Regeln* Prolog-spezifisch eingeben.

> "Da PROLOG über *Konstanten, Variablen* und *Strukturen* verfügt, kann man alle Probleme, die in einer anderen Programmiersprache geschrieben sind, auch in PROLOG beschreiben und lösen".
>
> (Maratschniger (1992a S.7ff.).

Der Aufbau eines TURBO-Prolog-Programmes gliedert sich in 4 [a]-fakultative Einheiten: *domains* (= [α-selbstdefinierte] Datentypen), *predicates*, *clauses* und ein *internes goal*

(= Abfragereglement, Programmsilhouette, -roundabout u.dgl., notwendig für das Kompilieren des Programmes).

Um den Einstieg in die NTS-Implementierungsumgebung zu erleichtern, betrachten wir zunächst ein kleines praktisches Anwendungsbeispiel:

predicates: computer(symbol,char,integer,real,symbol)

clauses: computer(pc,'386sx',345,1.44,berechnet_fraktale)

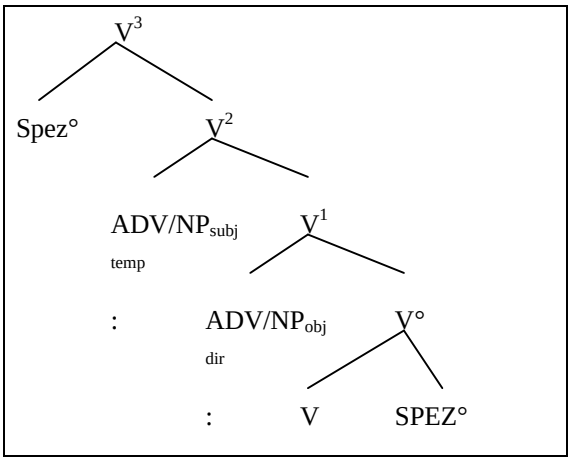
Die Angaben *Computerart, Prozessortyp, HD-Größe, Floppylaufwerk, Tätigkeit* definieren einen ganz bestimmten Computer eineindeutig. Die neben den *predicates* (= Satzaussagen) und *clauses* stehenden Zeilen stimmen in der Zahl der Elemente überein - die Bezeichnungen *symbol, char, integer, real* und *symbol* beschreiben die Kategorien, die in den clauses-Satz (= Wissensbasis) eingebettet sind.

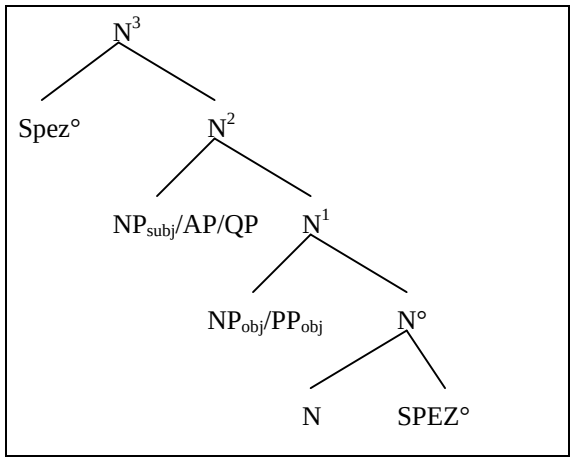
Die zugrundeliegende Struktur kann man nur dann verstehen, wenn man weiß, daß PROLOG eine Aussage als Einheit klassifiziert, die eindeutig wahr oder falsch sein kann. So wäre z.B. bezüglich der obigen Programmzeilen *Dieser PC hat eine 345MB HD* eine wahre Aussage, *Dieser PC hat eine 20MB HD* eine falsche. In beiden Fällen jedoch handelt es sich um die Beziehung zwischen einer bestimmten Computerart und deren HD-Kapazität.

Solche Beziehungen zwischen Objekten nennt PROLOG *Prädikate* und bettet sie in die folgende Syntax ein: *harddisk_von(pc,345)*, wobei gilt: *harddisk_von = Prädikat, pc = Objekt1* und *345 = Objekt2*.

3.0. Zur Implementierung der NTS in TURBO-Prolog

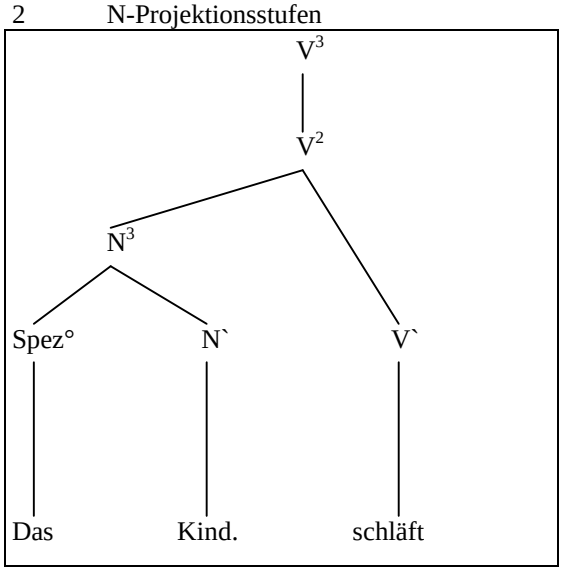
Der erste Schritt in der algorithmischen Erfassung des NTS-Regelapparates liegt in der Erstellung des *domains*-Teils. Jede Baumstruktur der NTS hat, wie bereits erwähnt, sofern sie einen spezifizierten Satz oder eine Nominalphrase darstellt, als X^{max}: X^3 (V^3, N^3). Die identische Projektionshöhe zeigt den parallelen Strukturaufbau beider Syntagmentypen:



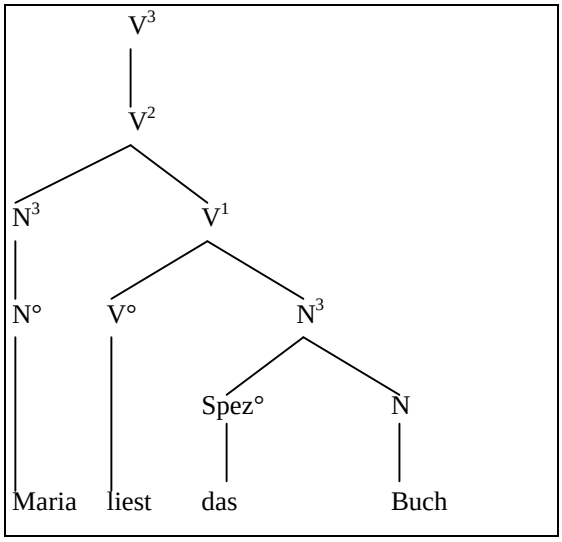


1 V-Projektionsstufen

2 N-Projektionsstufen



3 Beispielstruktur 3:
Das Kind schläft



4 Beispielstruktur 4:
Maria liest das Buch.

Der Ausgangspunkt einer jeden X^{max} ist also der *Keim*: X° (V°/N°). Er wird durch die Wortebene/Lexeme repräsentiert. Der in eine, vom Computer *a*utomatisch *g*enerierte, *K*lammerstruktur (AGK) überzuführende Satz besteht aus endlich vielen Einzellexemen (= Worten), die durch theoriespezifische Regeln miteinander verknüpft werden (Einzellexemkonkatination → Wortlisten). Aus dieser Überlegung folgt als erster domains-Eintrag: "Wortliste=Wort*".

Als Nächstes müssen die folgenden Fragestellungen domains-spezifisch kodiert werden:

1. Wie verzweigt die Maximalprojektion X^3?
2. Wie kann man "Varianten" darstellen, d.h. wie kodiert man das Faktum, daß einerseits N^3 durch pruning (vgl. (3)) sofort in N° übergeht (= z.B. Eigenname) aber sich andererseits bis N^3 entfaltet, wenn ein definiter Artikel das Nomen spezifiziert. Beide Male handelt es sich um eine N^3-Projektion; der Computer muß aber eindeutig zwischen den beiden unterscheiden können, um eine adäquate Klammerstruktur zu generieren. ⇒
3. Wie bildet man pruning ab? Die NTS verwendet aus Gründen der Ökonomie bei der Notation eine Stutzkonvention, die auf der Annahme basiert, daß jene Knoten, die nicht verzweigen, getilgt werden können. Ausnahmen hierzu bilden: 1. der oberste phrasale Knoten, also die durch $SPEZ^0$ bedingte Maximalprojektion (X^{max}) und 2. der Keim. Die X^{max} muß *immer* erhalten werden, weil sie die Fähigkeit eines lexikalischen Elementes oder einer Wortkette repräsentiert, eine syntaktische Funktion zu übernehmen. Man muß sogar annehmen, daß diese X^{max} (V^3/N^3) *immer* verzweigen, weil sie nicht nur eine kategorielle Projektionslinie, sondern zusätzlich eine Konstituente bei sich haben, die durch semantische Merkmale aufgefüllt ist (z.B.: modus-, aspektspezifizierende usw.).

Die X^{max} von A oder P müssen z.B. deshalb erhalten bleiben, weil lt. Theorie phrasale Kategorien *immer* anzuzeigen sind. Die Minimalprojektion (X° bzw. X^{min}) darf durch die Stützkonvention nicht getilgt werden, da sie Kopf der Kategorie ist.

Alle Entitäten, die einen P-Marker etablieren, stellen domains-spezifisch *string*-Einheiten dar und somit einen domains-Typ, der dazu geeignet ist, Wörter aufzunehmen, weil er sg. *Listen* (= Ketten/Verbindungen von Zeichen: hier Wörtern) verwalten kann. Somit erhalten wir als zweiten domains-Eintrag: *Wort=string*. Bevor wir nun die Strukturbäume unter (3) und (4) schrittweise in die domains-Schreibweise "verpacken", wollen wir uns zunächst einmal die entsprechenden Klammerstrukturen ansehen, also das, was der Computer im Endeffekt "selbständig" generieren soll:

ad (3): v3(v2(n3(spez0("Das"),n0("Kind")),v1(v0("schläft"))))

ad (4): v3(v2(n3(n0("Maria")),v1(v0("liest"),n3(spez0("das"),n0("Buch")))))

Um die Anweisungen zur Konstruktion solcher Klammerungen domains-spezifisch beschreiben zu können, muß streng zwischen Groß- und Kleinschreibung unterschieden werden. Kleingeschriebenes steht nur für Konstanten (= nicht veränderbare Größen) und Großgeschriebenes nur für Variablen (= veränderbare, weiter aufspaltbare Entitäten). Für die Implementierung bedeutet dies: Alle Projektionshöhen (= Knoten des P-Markers) sind Variablen, da sie in niedrigere Projektionen zerfallen und schließlich in Lexeme übergehen; die projektionshöhenanzeigende Notation vor den Klammern (= Klammertitel, z.B. v3 oder n0) nimmt den Status von Konstanten ein, da sie indexikalischen Charakter hat. Somit ergibt sich der folgende TURBO-Prolog-Eintrag für die Maximalprojektion V^3: V3=v3(V2). Diese Zeile drückt das Folgende aus:

1. V3 hat Variablenstatus weil es in V2 übergeht; auch V2 ist eine Variable, da der V^2-Knoten ebenfalls weiter verzweigt.
2. vn (wobei: n = 0,...,3) repräsentiert den Klammertitel. Nach dem gleichen Schema können wir nun die gesamte V-Projektion (= V-Kopflinie) des P-Markers unter (4) algorithmisch erfassen und dies sieht dann so aus: V3=v3(V2), V2=v2(N3,V1) und V1=v1(V0,N3).

Da wir mit X^0 (im Falle von V0) bereits die terminale Ebene erreicht haben, müssen wir in den domains noch festlegen, daß im Rahmen der AGK diese Keime mit Lexikonmaterial gefüllt werden. Darum schreiben wir: V0=v0(Wort) (wobei: Wort = Variable, in die hier z.B. Verben/Verbalformen eingesetzt werden). Somit haben wir alle Ebenen von V erfaßt. Nun gilt es, dies auch für die Kategorie N zu tun. In unseren Beispielstrukturen unter (3) und (4) werden wir mit [α spezifizierten] N^3-Projektionen konfrontiert, d.h. einmal liegt ein $SPEZ^0$ (hier: definiter Artikel) vor und einmal nicht (unter (4), = Eigenname → Anwendung von pruning, d.h. Tilgung von N^1 und N^2). Die domains-Programmierung ergibt somit für die spezifizierte NP: N3=n3(SPEZ0,N0) und für den Eigennamen unter (4): N3=n3(N0np) (wobei: np = *nomen proprium*) um den Spezifikator $SPEZ^0$, sowie dem Nomen N^0 bzw. N^0np (= Eigenname) auch einen Lexikoneintrag zuweisen zu können, müssen den domains

noch die folgenden Zeilen hinzugefügt werden: SPEZ0=spez0(Wort), N0=n0(Wort), N0np=n0np(Wort). Damit wäre der Satz unter (4) *Maria liest das Buch* im ersten Ansatz domains-spezifisch erfaßt.[2]

Aber was machen wir mit der Struktur unter (3)? Dem intransitiven Satz? Zunächst muß man sich vor Augen halten, was der Unterschied im Matrixverb strukturell auslöst: der transitive Satz weist zwei N^3 auf - eine Subjektsnominalphrase (NP_{subj}) und eine Objektsnominalphrase (NP_{obj}). Dem intransitiven Satz jedoch fehlt die NP_{obj}, da das Verb in seinem Thetaraster im unmarkierten Fall nur eine Thetarolle besitzt, der Subkategorisierungsrahmen daher keine Objektsnominalphrase aufweist.[3] Dieser Unterschied im Lexikon bedeutet programmiertechnisch, daß man in den domains eine Subkategorisierung von V^0 mittels Indizierung vornehmen muß: V0t steht dann für die *transitive Verbvariante* und V0i für die *intransitive*. Somit erhalten wir V2=v2a(N3,V0i) für die Struktur unter (3) und V2=v2b(N3,V1) für jene unter (4), wobei V1 in V0t und N3 verzweigt: V1=v1(V0t,N3). v2n (wobei: n = a,...,z sowie Kombinationen aus diesen) ist eine programmtechnische Notwendigkeit zur Variantenunterscheidung bzgl. der Verzweigungsmöglichkeiten ein und desselben Knotens, da für das System nur die Konstante *vor* der Klammer (hier: v2n) ausschlaggebend ist, nicht aber das, was *in* der Klammer steht, da es sich hierbei ohnehin nur um Variablen, wenn auch in einer NTS-abbildenden Notation, handelt. Um der semantosyntaktischen Struktur von TURBO-Prolog gerecht zu werden, müssen nun diese beiden V2-domains mit Hilfe des *logischen* oder (= ';') zu einem gemeinsamen *Aussageblock* zusammengeschlossen werden: V2=v2a(N3,V0i); v2b(N3,V1). Mit diesem Implementierungsvorgang haben wir zum Ausdruck gebracht, daß V^2 entweder als v2a oder als v2b realisiert werden kann.

Nun müssen wir uns aber noch einmal mit N^3 auseinandersetzen. Die domains-Zeilen, die N3 beinhalten, haben logisch-formal notiert die folgende Form: An=an(Xn,Yn) (wobei: n=a,...,z sowie Kombinationen aus diesen im Falle der Konstante a und n=0,...,3 im Falle der Variablen A,X,Y). Die N3 erscheinen für den Betrachter zunächst komplementär verteilt, d.h. einmal an der Stelle von X (in den domains-Zeilen für den V^2-Knoten) und das

---

[2] Die Kennzeichnung mit ***np*** ist notwendig, um dieses ***n0*** (den Eigennamen) von "anderen" ***n0*** zu unterscheiden; denn nur so kann programm(ier)technisch die Generierung von NP-Strukturen der Art *der Peter* vorerst verhindert werden. Ein sprachunabhängiger Parser muß schließlich aber auch in der Lage sein, Verbindungen von [definitem Artikel + Eigennamen] zu bearbeiten, weil er diese z.B. im Bairischen antrifft.

[3] Strukturen, wie z.B. *Der Mann träumt* vs. *Der Mann träumt einen schönen Traum*, also die Implementierungs-schritte bzgl. sg. *labiler Verben* stehen im Rahmen dieses Aufsatzes nicht zur Diskussion. Es sei nur kurz darauf hingewiesen, daß solche Verben zwischen semantischer und syntaktischer Valenz unterscheiden, wobei darauf geachtet werden muß, daß die syntaktische Valenz nie über der semantischen Valenz liegt und daher maximal nur gleich groß wie diese sein kann. Ein Verb wie *essen* könnte daher folgendermaßen klassifiziert werden: *essen1 (syntaktisch 1-stellig)* vs. *essen2 (semantisch 2-stellig)*. Die labile Valenz ist universell, weil sie der Gesprächsmaxime *"Be relevant!"* (Grice) gehorcht.

andere Mal an der von Y (in der domains-Zeile für den V^1-Knoten). Würden wir nun unser Lexikon mit "neutralen" N^0 (und $SPEZ^0$) auffüllen (= N^0 (und $SPEZ^0$) ohne Setzung von Indizes bzgl. der Kongruenzmerkmale), erhielten wir vom Computer auch Klammerstrukturen für Sätze wie: *Dem Mann ißt der Apfel* oder *Peter liest die Buch*. Betrachten wir also die NP_{obj} in der domains-Zeile für den V^1-Knoten etwas genauer: Der bestehende Eintrag würde die Erzeugung einer Struktur (AGK) für ... l*iest das Buch* genauso zulassen wie für ... *liebt die Musik*. Da im Deutschen Nominativ und Akkusativ bei Neutra und Femina formal gleich kodiert werden, fällt das eigentliche Problem, um das es hier geht, zunächst nicht gleich auf; im Falle der Maskulina, also bei Strukturen wie ... *ißt der Apfel* - für die wir ebenfalls eine AGK erhalten würden - wird aber klar, daß eine effiziente Implementierung der NP_{obj} nur auf eine programminterne Indizierung bzgl. *strukturellem* und *obliquem* Kasus hinauslaufen kann. Wir setzen daher für V^1 eine leicht modifizierte domains-Zeile an: V1=v1a(V0t,N3akk) (wobei: akk = Akkusativ). Daraus ergibt sich nun als notwendige bzw. zwingende Konsequenz auch eine Erweiterungsdefinition bzgl. der N3-domains-Zeilen, d.h. die NP_{obj} muß sowohl extern (= Konstante) als auch intern (= Variablen) mit einen Kasusindex akk versehen werden, was dann so aussieht: N3akk = n3akk (SPEZ0akk, N0akk), (wobei: n3akk = Konstante, SPEZ0akk und N0akk = Variablen). Damit ist der erste entscheidende Schritt in Bezug auf *Kongruenzimplementierung*, zunächst einmal im Bereich Kasus, getan.[4] Die beiden NP_{subj} N3=n3a(SPEZ0,N0) und N3=n3b(N0np) müssen keiner Änderungen unterzogen werden, da die NTS den Nominativ als "Nicht-Kasus" klassifiziert, was in eine "Nicht-Kasusindizierung" (mit z.B. nom für Nominativ) resultiert.

Das Einzige, was in dieser Programmebene jetzt noch fehlt ist die domains-Erfassung von SPEZ0akk und N0akk - lexikalische Kategorien bzw. terminale Knoten, die wie folgt zu notieren sind: SPEZ0akk=spez0akk(Wort) und N0akk=n0akk(Wort).[5]

Der nächste Schritt in der Generierung unseres NTS-Parsers ist die Programmierung der *predicates*-Zeilen, denn im clauses-Teil können später nur solche Verbindungen auftreten, deren Elemente eindeutig in den predicates vereinbart wurden; sie greifen noch einmal auf *alle* domains-Größen zu und bereiten sie auf die, für die AGK notwendige, *Listenver*- und -

---

[4] Wie komplex die programmiertechnische Aufarbeitung von Kongruenz ganz allgemein gesehen ist, sollte klar geworden sein. Der gesamte Vorgang der Kongruenzimplementierung kann hier aus Platzgründen nicht beschrieben werden. Er besteht, generell gesprochen, aus der Etablierung einer gezielt und gut überlegten Indizierungsregularität, die sich mit Hilfe programmtechnischer Feinheiten fast nur mehr im Lexikon abspielt und Teil der algorithmischen Erfassung der Semantik (im Rahmen der NTS) ist. Vgl. Sie hierzu auch Maratschniger (1993c), (1994a) und (1994c).

[5] Bereits durch den Einsatz von *Kasusindizierung* wird es möglich, Sätze aus anderen Sprachen zu analysieren, unabhängig davon ob sie Kasus [α morphologisch] kodieren oder nicht. Vgl. Sie Englisch, wo Kasus *über die Stellung im Satz* ausgedrückt werden kann mit Russisch oder Polnisch, wo eine *morphologische Kodierung* stattfindet.

bearbeitung vor. Die Liste - sie kann [α leer] sein - ist die rekursive Basisdatenstruktur von PROLOG-Programmen. Mit ihrer Hilfe und einem in den clauses lokalisierten Satzzerlegungs- und Morphem-/Lexemkonkatinations-Mechanismus der hier aus Platzgründen nicht näher beschrieben werden kann, können wir dem Computer einen Satz ($S^{(\prime)}$ bzw. V^3, wobei: ' = spezifizierter Satz) eingeben und erhalten dann von ihm als Ausgabe eine NTS-getreue, P-Marker adäquate, Klammerstruktur (AGK). [6], Wie sieht nun so eine predicates-Zeile aus? z.B. so: v3(V3,Wortliste,Wortliste); zu lesen wäre sie als eine "Anweisung", die die Verknüpfung von Einzellexemen innerhalb von bzw. zu Wortlisten (= Variablen) zur Folge hat, die den späteren inneren Aufbau der Variablen V3 und somit des Satzes ergibt; die Konstante v3 schließlich, benennt genau die Klammer eindeutig, in der sich das soeben Beschriebene abspielt. n0np(N0np,Wortliste,Wortliste), v2a(V2,Wortliste,Wortliste) wären z.B. weitere predicates-Zeilen, die sich aus unseren domains ergeben.

Der weitaus wichtigste Teil des NTS-TURBO-Prolog-Parsers ist jedoch die *clauses*-Ebene, denn erst hier werden die "Erkenntnisse" aus den domains und predicates durch den Regelapparat der NTS zusammengeführt und erst hier findet auch die Etablierung der Lexikoneintragungen sprich des Lexikons selber statt. In unserem Fall bestehen die clauses nur aus Regeln, also aus Prädikaten, die zwei Ebenen umfassen: *Regelkopf* und *Regelkörper*, verbunden durch das *logische wenn/dann* (= ':-' entspricht hier dem '→' von Transformationsregeln). In den clauses-Teil fallen auch alle Bedingungen für die hardwaremäßige Aufnahme eines Satzes, dessen softwaremäßige Verarbeitung, Transformation in und Ausgabe *als* Klammerstruktur(en).

Damit sich aber nun endlich die NTS-adäquaten AGKs in Bezug auf unsere Beispielsätze unter (3) und (4) "aus dem Computer erheben (können)", müssen wir folgendermaßen vorgehen: Für V3 gibt es in den domains und predicates diese beiden Einträge: V3=v3(V2) und v3(V3,Wortliste,Wortliste). Indem wir nun die domains in die predicates einsetzen und statt Wortliste - LX schreiben (wobei: X=1,2,3 in Bezug auf die Implementierung von binären Strukturbäumen) erstellen wir die gewünschte V3-clauses-Zeile. Diese LX geben nun "den Weg" für das Konkatinieren der terminalen Elemente (= der einzelnen Wörter) vor. Wir erhalten somit für den NTS-Knoten V^3 (= spezifizierter Satz) den folgenden clauses-Ausdruck: v3(v3(V2),L1,L2) (= Regelkopf). Er geht dann in weiterer Folge mit Hilfe der bereits erwähnten logischen wenn/dann-Bedingung (besser: Transformationsbedingung) in eine Aufgliederung des NTS-Knotens V^2 über (=

---

[6] Die grundlegenden Programmierschritte (clauses) hierzu sehen wie folgt aus: (die Übereinstimmung bzgl. domains, predicates und internem goal soll und kann im Rahmen dieses Aufsatzes nicht dokumentiert werden. Näheres dazu in Maratschniger (1992a) und (1993b)):
readlist("Satz"):-readln(Satz),write_list(Satz,Anfang),v3(Satzstruktur,Anfang,[ ]),write(Satzstruktur).

write_list(Satz,[Wort|Restwortliste]):-fronttoken(Satz,Wort,Restsatz),write_list(Restsatz,Restwortliste).
write_list(X,[]):- write("\n"),X=".";X="".

Regelkörper), die sich erneut aus dem Zusammenspiel der entsprechenden V2-domains und -predicates ergibt. Das Resultat ist schließlich unsere *allererste* NTS-Ersetzungsregel in TURBO-Prolog-Manier:

v3(v3(V2),L1,L2):-v2a(V2,L1,L2);v2b(V2,L1,L2).

Als Nächstes müssen die v2-Varianten "clausifiziert" werden. Dies erfolgt mittels der beiden v3-Regelkörper. Die erste Struktur, die sich aus dem Einsetzen der domains in die predicates ergibt ist v2a(v2a(N3,V0i),L1,L3) und die *zweite*: v2b(v2b(N3,V1),L1,L3). Diese beiden Einträge repräsentieren Regelköpfe. In Verbindung mit den entsprechenden Regelkörpern ergibt sich dann für den intransitiven Satz *Der Mann schläft* (= (3)) diese NTS-adäquate clauses-Zeile: v2a(v2a(N3,V0i),L1,L3):-n3a(N3,L1,L2),v0i(V0i,L2,L3). und für den transitiven Satz *Peter liest das Buch* (= (4)): erhalten wir dementsprechend: v2b(v2b(N3,V1),L1,L3):-n3b(N3,L1,L2),v1(V1,L2,L3). Jetzt fehlen nur noch die entsprechenden clauses-Einträge für:

a. den VP-Knoten V1:
   v1(v1(V0t,N3akk),L1,L3):-v0t(V0t,L1,L2,),n3akk(N3akk,L2,L3).
b. die spezifisch *kontextvariierte* N3-Projektion:
   1. (für die NP_{subj}):
   a. n3a(n3a(SPEZ0,N0),L1,L3):-spez0(SPEZ0,L1,L2),n0(N0,L2,L3).
   b n3b(n3b(N0np),L1,L2):-n0np(N0np,L1,L2).
   2. (für die NP_{obj}):
   n3akk(n3akk(SPEZ0akk,N0akk),L1,L3):-spez0akk(SPEZ0akk,L1,L2),n0akk(N0akk,L2,L3
)..

Den Abschluß unseres Implementierungsansatzes bildet die "clausale" Etablierung des Lexikonabschnittes, also jener Ebene, die die Minimalprojektionen (Keime) mit Lexemen anfüllt. Hier werden die einzelnen terminalen Einträge der Regelkörper widergespiegelt und es kommt zu einer spezifischen internen Aufgliederung der Liste in ihre Elemente *Kopf* und *Rumpf*, um das Gelingen der Lexemkonkatination zu gewährleisten - eine detaillierte Beschreibung des programmtechnischen Hintergrundes von Ursache, Nutzen und Wirkung dieser Listenhandhabung kann und soll hier nicht näher erläutert werden (Vgl. hierzu Maratschniger 1992a und 1993b). Das für unsere beiden Beispielsätze unter (3) und (4) notwendige Lexeminventar sieht dann so aus:

n0(n0(N0),[N0|L],L):-N0=Mann. n0np(n0np(N0np),[N0np|L],L):-N0np=Peter.
n0akk(n0akk(N0akk),[N0akk|L],L):-N0akk=Buch.
spez0(spez0(SPEZ0),[SPEZ0|L],L):-SPEZ0=der.
spez0akk(spez0akk(SPEZ0akk),[SPEZ0akk|L],L):-SPEZ0akk=das.
v0i(v0i(V0i),[V0i|L],L):-V0i=schläft. v0t(v0t(V0t),[V0t|L],L):-V0t=liest.

ZUSAMMENFASSUNG

Mit Hilfe des hier vorgestellten Programmierkonzeptes konnten bisher *alle* Teilbereiche der NTS mehr als nur zufriedenstellend implementiert werden: inkohärente Infinitivkonstruktionen genauso wie morphosyntaktische Analysen mittels X^{-1}, lexikalisch nicht gefüllte Elemente (z.B. "klein pro" oder "groß PRO") ebenso wie die wenigen theoriebedingten ternären Verzweigungen. Der NTS-Parser ist bereits so weit entwickelt, daß er auf bestimmte grammatisch und/oder semantisch falsche Eingaben mittels Fehleranalyse antwortet, also nicht nur erkennt, daß Sätze wie 1.) *Der Mann liest den Buch*, 2.) *Die Katze trinkt die Butter* und 3.) *Das Pferd liest den Buch* einfach falsch sind, sondern auch die Fehlerquelle lokalisiert und an den User weitergibt - also im Falle von 1.) einen Kongruenzfehler (KF) in der Objekts-NP meldet, bzgl. 2.) semantische Inkompatibilität (SIK) von Matrixverb und Objekts-NP signalisiert und 3.) als eine Kombination aus SIK (bzgl. Matrixverb und Subjekts-NP) und KF (innerhalb der Objekts-NP) charakterisiert.

VERZEICHNIS DER ABKÜRZUNGEN UND SONDERZEICHEN

NTS = *N*atürlichkeits*t*heoretische *S*yntax
NT = *N*atürlichkeits*t*heorie
AGK = *a*utomatisch *g*enerierte *K*lammerstruktur
KF = *K*ongruenz*f*ehler
SIK = semantische *Ink*ompatibilität
a = +/-
® = daraus entsteht/daraus ergibt sich
Þ = daraus folgt
TURBO-Prolog bezogen:
, = logisches und
; = logisches oder
:- = logisches wenn/dann

LITERATUR

Abraham, W. 1982 *Wortstellung und das Mittelfeld im Deutschen*, Groningen (Ms).

Bratko, I. 1986 *Prolog Programming and Artificial Intelligence*, Addison-Wesley.

Chomsky, N. 1981 *Lectures on Government and Binding*, Dordrecht.

Dressler, W.U./Mayerthaler, W./Panagl, O./Wurzel, W.U. 1987 *Leitmotifs in natural Morphology*, Amsterdam.

Fliedl, G. 1988 Kontrollphänomene und thematische Rollen, in: *2. Jenaer Semantik-Syntax-Symposium*, Jena: 135-140.

Haider, H. 1984 Was zu haben ist und was zu sein hat: Bemerkungen zum Infinitiv, *Papiere zur Linguistik* 30: 23-36

Maratschniger, M. 1992a *Zur Implementierung der NTS ("NatürlichkeitstheoretischeSyntax") in TURBO-Prolog* (Dissertation Univ. Klagenfurt).

1992b Nichts ist ordentlicher als das Chaos !?, *Scientia* 32, Innsbruck.

1993a Statistische und kartographische Repräsentation der Infinitivprominenz in (nord)italienischen Dialekten, *Scientia* 40, Innsbruck.

1993b *Algorithmen zur Implementierung von Teilbereichen der NTS ("Natürlichkeitstheoretische Syntax") und NTMS ("Natürlichkeitstheoretische Morphosyntax") in PDC- und TURBO-Prolog* (Projektbericht 1, Institut für Sprachwissenschaft der Univ. Klagenfurt).

1993c *Vom Satz zur NTS-Baumstruktur. Vom Wort zur NTMS-Baumstruktur. Eine computerlinguistische Studie anhand von TURBO-Prolog - PDC-Prolog - ICON - Word Perfect 5.1* (Projektbericht 2, Institut für Sprachwissenschaft der Univ. Klagenfurt).

1993d Gibt es einen slawischen Einfluß in Norditalien? Kurz-Statistische und Teil-Kartographische Aufarbeitung der Infinitivprominenz in ausgewählten slawischen und nordöstlich angrenzenden norditalienischen Sprachproben, *Klagenfurter Beiträge zur Sprachwissenschaft*, 17-18: 173-199.

1993e Gibt es einen slawischen Einfluß in Norditalien? Kurz-Statistische und Teil-Kartographische Aufarbeitung der Infinitivprominenz in ausgewählten slawischen und nordöstlich angrenzenden norditalienischen Sprachproben, *Acta Linguistica Hafniensia* 26: 67-79.

1994a Die Syntax dem Computer - Die Semantik dem Menschen? Eine computerlinguistische Studie anhand des NT(M)S-TURBO-Prolog Parsers. Erscheint im Rahmen der Publikationen zu den 4. Münchner Linguistik Tagen vom 14.-16.3.1994).

1994b Kontrollrelationen und ihre Implementierung in TURBO-Prolog. Eine computerlinguistische Studie, *Grazer Linguistische Studien* 41: 41-52.

1994c Das Pferd liest das Brot. oder: Zur Rolle der Semantik im Rahmen der Implementierung der NTS (= Natürlichkeitstheoretische Syntax) in TURBO-Prolog, in: Halwachs, D.W./Stütz, I. (Hgg.) Sprache - Sprechen - Handeln. Akten des 28. Linguistischen Kolloquiums Graz 1993, Tübingen: 333-340.

Maratschniger, M./Fliedl, G. 1991 Natürlichkeitstheoretische Syntax - Ein Modell und seine Anwendung, in: Klein, E./Pouradier-Duteil, F./Wagner, K.H. (Hgg.) *Betriebslinguistik und Linguistikbetrieb. Akten des 24. Linguistischen Kolloquiums, Universität Bremen, 4.-6. September 1989*. Band 1, Tübingen: 281-295.

1993 Zur Implementierung der "LEEREN" Subjekte "PRO" und "pro" in Turbo-PROLOG im Rahmen der NTS ("Natürlichkeitstheoretische Syntax"), *Papiere zur Linguistik* 48/1: 73-88.

Mayerthaler,W./Fliedl,G. 1993 Natürlichkeitstheoretische Syntax (NTS), in: Jacobs, J../v. Stechow, A./ Sternefeld, W./Vennemann, Th.. (Hgg..) *Ein internationalesHandbuch zeit-genössischer Forschung. (An international Handbook of Contemporary Research)*, Berlin/New York:

PDC Prolog Version 3.30 1992 *User's Guide. Prolog Development Center A/S H.J,* Copenhagen.

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

Stechow, A./Sternefeld, W. 1988 *Bausteine syntaktischen Wissens. Ein Lehrbuch der generativen Grammatik*, Opladen.

Martina Maratschniger
Institut für Sprachwissenschaft der Universität Klagenfurt