Gefördert vom

bmb+f

Bundesministerium für
Bildung, Wissenschaft,
Forschung und Technologie

Verbmobil

Verbundvorhaben

Die japanische Syntax im Verbmobil Forschungsprototypen

Melanie Siegel

DFKI

Vm

Report 133
Juni 1996

Juni 1996

Melanie Siegel

Deutsches Forschungszentrum
für Künstliche Intelligenz GmbH
Stuhlsatzenhausweg 3
66123 Saarbrücken

Tel.: (0681) 302 - 5284
Fax: (0681) 302 - 5341
e-mail: `siegel@dfki.uni-sb.de`

Gehört zum Antragsabschnitt: 6.2 Japanische Syntax

Die vorliegende Arbeit wurde im Rahmen des Verbundvorhabens Verbmobil vom Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF) unter dem Förderkennzeichen 01 IV 101 G gefördert. Die Verantwortung für den Inhalt dieser Arbeit liegt bei der Autorin.

1 Einleitung

VERBMOBIL setzt für die Implementierung der japanischen Syntaxbeschreibung im gegenwärtigen Forschungsprototypen den Grammatikformalismus TrUG ([CS94], [BS92], [BS94]) ein. TrUG ist ein unifikationsbasierter Grammatikformalismus mit Phrasenstrukturregeln und Merkmalsannotationen. Eine TrUG-Grammatik besteht aus Syntax, Lexikon, Deklarationsfile und Makros. Die syntaktischen Grammatikregeln sind kontextfreie Phrasenstrukturregeln mit Merkmalen und Pfadgleichungen. Der Deklarationsfile definiert die Wertebereiche. Makros erlauben Abkürzungen von Feature-Struktur-Gleichungen. Die Grammatik ist derart formuliert, daß sie mit HPSG prinzipiell kompatibel ist. Syntax und Lexikonbeispiele sind im Anhang dokumentiert. Der Semantikformalismus ist in [BMMM94] beschrieben.

Die Domäne in VERBMOBIL sind Terminaushandlungsdialoge. Für die Syntax bedeutet das zunächst, daß die Syntax sich an gesprochener Sprache orientieren muß. Das beinhaltet Nullanaphern, Phrasen, die auf die Kommunikationssituation bezogen sind und Phrasen, die für geschriebene Sprache als nicht wohlgeformt bezeichnet werden. Weitergehend gibt es einige domänenspezifische syntaktische Besonderheiten, wie zum Beispiel die Realisierung von Zeitangaben.

2 Die japanische Syntax

Japanisch wird als Head-Final-Sprache betrachtet. So steht – in Haupt- wie in Nebensätzen – das Verb immer hinter seinen Komplementen. Nominalphrasen werden (normalerweise) mit Postpositionen annotiert. Die Funktionen dieser Postpositionen können Topikalisierung, Kasusmarkierung (zur Argumentselektion durch das Verb) oder eine der deutschen Präposition entsprechende Funktion sein. Daher wird diskutiert, ob zwischen Kasuspartikeln mit rein syntaktischen Funktionen und Postpositionen mit syntaktisch-semantischen Funktionen unterschieden werden muß. Dabei ist die Grenzziehung nicht immer eindeutig: Die Postposition *ni* kann Kasusfunktion (Dativ) oder auch semantische Funktion (als direktionale Präposition) haben.

[Gun87] betrachtet die Postpositionen als eine gemeinsame syntaktische Klasse, die Head Phrase ist, die aus NP und Partikel besteht. Diese Phrase nennt er PP. [Miy86] vertritt jedoch die Position, daß zwischen Postpositionalphrasen und Nominalphrasen unterschieden werden muß wobei letztere Kasuspartikel enthalten. Auf die Argumentation soll an dieser Stelle im Einzelnen nicht eingegangen werden. Die Betrachtungsweise, Postpositionen uniform zu kategorisieren und generell als Kopf einer Phrase anzusehen, macht es möglich, von einem generellen Schema für Syntaxregeln auszugehen, bei dem jeder Mutterknoten aus einem Tochterknoten und einem Kopfknoten besteht. Die Unterscheidung von Adjunkten und Komplementen geschieht dann über die Verbvalenz gekoppelt mit der postpositionalen Information.

Ein Problem der japanischen Syntax ist die freie Wortstellung von Komplementen und Adjunkten im Satz. Dies hat dazu geführt, daß eine Diskussion entstand, ob der Phrasenstrukturbaum flach (d.h. ohne VP) oder hierarchisch (d.h. mit VP) organisiert ist[1]. Wir haben uns für eine hierarchische Analyse entschieden, da der VP-Knoten für VP-Komplemente und für die Analyse von Reflexivpronomina benötigt wird (Für die weitere Argumentation siehe [Gun87]).

[1] Für eine flache Analyse siehe [Far84], für eine hierarchische Analyse siehe [Gun87]

Ein weiteres Problem ist, daß vom Verb subkategorisierte Argumente an der Satzoberfläche fehlen können, wie zum Beispiel das fehlende Subjekt in:

ikaga	deshoo	ka
gut	COP	QUE

Eine Analyse, bei der dieses Problem vom Aufbau der Phrasenstruktur unabhängig auf der Ebene des Subkategorisierungsrahmens gelöst wird, ist in der derzeitigen Konstellation in Verbmobil nicht realisierbar. Daher muß die Möglichkeit zugelassen werden, daß PPs leere Knoten sind. Nicht ganz einfach ist es, diese Option wiederum so zu beschränken, daß die Anzahl der möglichen Lesarten für einen Satz tolerierbar bleibt.

Japanische Nomen enthalten keine Informationen über Numerus, Person und Definitheit. Die Verben flektieren daher nicht - wie zum Beispiel im Deutschen - nach Person und Numerus des Subjekts und geben auch keine syntaktischen Hinweise auf die Art eines fehlenden Komplements.

Topikalisierung von PPs ist im Japanischen syntaktisch durch die Postposition *wa* markiert. Dabei kann ein Verbargument Topik werden, es können aber auch Topik-Adjunkte auftreten:

nanimo	yotee	wa	ima	no	tokoro	hait
irgend ein	Plan	TOPIC	jetzt	GEN	Zeitpunkt	eingefügt
te	ori	maseN				
TE	AUX	HON,NEG				

(Zu diesem Zeitpunkt habe ich noch keine Termine)

juu	ichi	nichi	no	hi	wa	seminaa	ga
10	1	Tag	GEN	Tag	TOPIC	Seminar	GA
ichi	nichi	juu	hait	te	ori	mashi	
1	Tag	während	eingefügt	TE	AUX	HON	
te							
TE							

(Am 11. gibt es den ganzen Tag lang ein Seminar)

3 Die japanischen Dialoge in Verbmobil

Ein wesentlicher Unterschied der Verarbeitung von Dialogen zur Verarbeitung von Einzelsätzen oder Texten ist, daß sie aus Turns bestehen. Die Eigenschaften von Turns im Gegensatz zu (geschriebenen) Sätzen sind:

1) Sie können Sequenzen von Segmenten sein.

2) Sie können fragmentarisch sein.

3) Sie können syntaktischen Regeln unterliegen, die für die geschriebene Sprache nicht gültig sind.

Dabei steht der Begriff 'Segment' für eine Äußerungseinheit in der gesprochenen Sprache, parallel zu der Äußerungseinheit der geschriebenen Sprache, dem Satz.

Dialoge enthalten Turns, die unter Umständen zur weiteren Verarbeitung segmentiert werden müssen. Anders als bei der deutschen Analyse in Verbmobil gibt es für die japanische Analyse nicht das Hilfsmittel der Prosodieerkennung. Die Syntax hat also die Aufgabe, nach syntaktischen und lexikalischen Merkmalen zur Erkennung von Segmentgrenzen zu suchen, ohne Zugang zu prosodischer Information zu haben.

Ein ganz wesentliches Merkmal japanischer Dialogsprache ist der Hörerbezug. Durch Formalitätsmerkmale der Sprache werden die soziale Relation zwischen Sprecher und Hörer und die soziale Relation zwischen Sprecher und Subjekt ausgedrückt. Hier sind verbale Flexionen (*shi masu*), honorifische Präfixe (*o genki*) und Auxiliare (*hait te ori masu*) betroffen. Eine Syntax, die japanische Dialoge analysieren soll, muß solche Konstruktionen verarbeiten und die Information weitergeben können.

Anders als in Texten kommt es in japanischen Dialogen vor, daß Postpositionen weggelassen werden, wie im folgenden Beispiel:

hai	tokuni	atakushi	$\emptyset$	kono	hi
ja	besonders	ich	PARTIKEL	dieser	Tag

$\emptyset$	yotee	ga	ire	te	ori	maseN
PARTIKEL	Plan	GA	eingefügt	TE	AUX	HON,NEG

ja, an diesem Tag habe ich keine besonderen Pläne

Da diese jedoch zur Identifizierung von Verbargumenten und Adjunkten dienen, führt dies zu Problemen. Einerseits müssen Ellipsen von Postpositionen zugelassen werden, andererseits muß diese Option so beschränkt werden, daß die Anzahl der Lesarten für jeden Satz in einem tolerierbaren Maß bleibt.

Da es sich beim Verbmobil-Szenario um Terminabsprachen handelt, muß eine Syntax für Zeitausdrücke, wie zum Beispiel *juu shichi nichi kayoobi* – Dienstag der 17. oder *raigetsu no futsuka* – der zweite im nächsten Monat, erstellt werden.

Ein Problembereich betrifft die Wortsegmentierung und die Morphologie. Da in der japanischen Schriftsprache keine Abstände zwischen den Wörtern stehen, gibt es keine allgemein anerkannte Einigung darüber, was genau Wörter sind. Der Output der Spracherkennungskomponente liefert zum Beispiel Verbstämme und Verbflexionen teilweise als eigene Wörter. Daher muß die Syntax auch Morphologieregeln enthalten, um diese 'Wörter' zusammenzufassen.

4 Allgemeine Prinzipien

Die japanische Syntaxentwicklung in Verbmobil folgt einigen allgemeinen Prinzipien, die im folgenden dargelegt werden sollen.

Da Japanisch eine Head-Final-Sprache ist, folgen die Syntaxregeln dem Prinzip der binären Verzweigung, wie sie von [Gun87] in der grundlegenden Syntaxregel formuliert sind:

$$M \rightarrow D \quad H$$

Jeder Mutterknoten (M) expandiert zu einem Tochterknoten (D) und einem Kopfknoten (H). Die Beziehung zwischen Mutterknoten, Tochterknoten und Kopfknoten wird in der JPSG durch Prinzipien bestimmt, die an die Prinzipien der HPSG angelehnt sind ([Gun87], S.223):

HEAD Feature Principle:

HEAD(M) unifies with HEAD(H), where HEAD(C) is the set of HEAD features of category C[2].

SUBCAT Feature Principle:

One of the following must hold:

Complementation:	*SUBCAT(H) unifies with SUBCAT(M), except for the category that unifies with D*
Adjunction:	*SUBCAT(M) unifies with SUBCAT(H)*
Coordination:	*SUBCAT(M),SUBCAT(D), AND SUBCAT(H) unify*

Auch in der für VERBMOBIL entwickelten Grammatik sind die Phrasenstrukturregeln binär verzweigt. Allerdings gibt es - bedingt durch die Anforderungen des Parsers TrUG - explizite Regeln wie $VP \longrightarrow PP \;\; VP$ und $PP \longrightarrow NP \;\; P$, und nicht nur eine allgemeine Regel mit Variablen wie bei Gunji (siehe Anhang). In einem HPSG-basierten Ansatz ließe sich dies mit Typhierarchien realisieren. Die Prinzipien werden im TrUG-Formalismus durch Makros und Pfadgleichungen in den Regeln realisiert.

Es gibt Head-Features und ein Subcat-Feature. Head-Features von verbalen Projektionen sind Finitheit, Tempus, Modus und Formalität[3]. Head-Features von Postpositions-Projektionen sind Kasus und *empty* (für fehlende Komplemente). Das Subcat-Feature enthält Information über die subkategorisierten Komplemente.

Das Makro *hfp* (=head feature principle, siehe Anhang 'Makros') sorgt für die Vererbung der Head-Features vom Kopfknoten an den Mutterknoten. Das Makro *complement* (siehe Anhang) bewirkt die Unifizierung des Tochterknotens mit einem Element aus dem Subkategorisierungsrahmen des Kopfknotens und für die Weitergabe der Information an den Mutterknoten. *adjunct* sorgt für die Eintragung eines Tochterknotens als Adjunkt und die Vererbung der Subkategorisierungs- und der Head-Information des Kopf-Knotens an den Mutterknoten.

Während frühere Ansätze (z.B. [Kun73] Transformationen für die Beschreibung der japanischen Sprache vor allem wegen Scrambling-Phänomenen und Topikalisierung als unabläßlich ansahen, gehen neuere Ansätze der Unifikationsgrammatiken (z.B. JPSG, HPSG, LFG etc.) davon aus, daß die Einführung von Features im Zusammenhang mit einer semantischen Darstellung Transformationen ersetzen können. Auch der in Verbmobil verfolgte Ansatz für die syntaktische Analyse des Japanischen ist nichttransformationell. Argumente des Verbs werden durch ihre Kasuspartikel identifiziert, und nicht durch ihre Stellung im Phrasenstrukturbaum. In TrUG besteht die Möglichkeit der Einführung syntaktischer Merkmale und der semantischen Interpretation.

[2] Hier ist das Head Feature Principle für den Fall formuliert, daß der Tochterknoten ein Komplement (C) ist.

[3] Englisch: Honorifics

5 Das Lexikon

5.1 Nicht-verbale Kategorien

Die Kategorien, die in das Lexikon aufgenommen wurden, werden im folgenden aufgelistet.

- Nomen u.a.

  Für Zeitausdrücke stehen im Lexikon Zahlen, temporale Modifikatoren (*han, sugi...*), numerale Klassifikatoren und Datumsangaben. Nomen enthalten weder Numerus- noch Kasus- noch Genusinformation. Als Nominalisierungen sind *no*, *N*, *katachi* und *koto* möglich. Für Eigennamen sind Personennamen, Titel, Institutionsnamen und Pronomina vorgesehen. Eine weitere Kategorie sind spezielle Nomen, die ohne Postposition auftreten können. Würde man das bei allen Nomen zulassen, so bekäme man ein Problem mit der hohen Anzahl der Lesarten in vielen Fällen. Betrachtet man allerdings die Dialogdaten, so kann man feststellen, daß es eine wiederkehrende Menge von Nomen ist, die häufig ohne Postposition auftreten. Aufgenommen wurden bisher *tokoro, tsugoo, mooshiwake, hiru* und *jikan*. Nomen können von Determinatoren begleitet werden.

- Postpositionen

  Postpositionen sind uniform kategorisiert, es wird kein kategorieller Unterschied zwischen Kasuspartikeln und Postpositionen mit semantischen Funktionen gemacht.

  Die Postpositionen haben ein Kasuswert und einen Wert für die Realisierung (als fehlende oder realisierte Postposition), die an die PP hochgereicht werden.

- Satzendepartikel

  Satzendepartikel haben häufig pragmatische Funktionen. Sie können zum Beispiel Emphase oder Abschwächung ausdrücken. Die Satzendepartikel *ka* macht einen Aussagesatz zu einem Fragesatz.

- Adjektive

  Es ist notwendig, zwischen *i-Adjektiven* und *na-Adjektiven* zu unterscheiden. So kann ein na-Adjektiv in einer Adverbialphrase mit einer Postposition auftreten (*genki de*:

hai	okagesama	de	genki	de	yat	te	ori	masu
ja	Dank	DE	gesund	DE	leben	TE	HON	HON

  ja, vielen Dank, es geht mir gut

  Beide Adjektivarten können ein honorifisches Präfix haben (*o genki, o isogashii*, HON-gesund, HON-beschäftigt) und auch modifiziert werden(*moo sorosoro*, schon bald). Na-Adjektive brauchen ein *na*-Suffix, wenn sie ein Nomen modifizieren. Dieses ist in den Transkriptionen - und damit auch im Lexikon - ein eigenes Wort.

- Interjektionen

  Zum Beispiel *hai, iie, dewa, jaa*

- Adverbien

  Zum Beispiel *dake, mochiron, itsumo*

- QUOT

  Für das Wort *yuu* gibt es wegen seiner speziellen Verwendungsweise eine eigene Kategorie.

- Konjunktionen

  Es gibt die Satzkonjunktionen *to* und *shi* und die Nominalkonjunktionen *to* und *mata*.

- Idiome

  Es ist m.E. der effektivste Weg, Idiome als solche zu erkennen und zu übersetzen, ohne eine weitere Analyse vorzunehmen. Daher stehen sie unter dem Eintrag *idiom* im Lexikon.

5.2 Verben

Es gibt die Unterscheidung zwischen Head-Features und Subkategorisierungs-Features. Head-Features enthalten folgende Merkmale:

- Finit: Ja oder Nein.
- Modus: Indikativ, Konditional, Negativ, Frage, Imperativ, Voluntativ, Potentialis.
- Tempus: Present, Past, no_tense (für die Te-Form), Future, Progressive.
- Form: Plain, Hon.

Subkategorisierungsfeatures enthalten Merkmale für die subkategorisierten Funktionen, SUBJ:case, OBJ:case, OBJ2:case. Kasuswerte sind die Postpositionen und *no_case* für subkategorisierte Funktionen, die keine Kasusmarkierung brauchen. Ein transitives Verb hat also im Lexikoneintrag ein Subjekt mit (z.B.) Kasus *ga* und ein Objekt mit (z.B.) Kasus *o*. Es gibt folgende Verbarten:

- Auxiliare

  Es gibt (bedingt durch die Transkriptionen) Vollformen, die alle Head-Features enthalten und Stämme, die keine Features enthalten, da Subkategorisierungsinformation nicht von den Auxiliaren kommt.

- Kopula

  Es gibt auch hier Vollformen, die Head- und Subkategorisierungsfeatures enthalten und Stämme, die Subkategorisierungsfeatures enthalten.

- Suru

  Eine besondere Klasse von Verben kann mit einem Nomen auftreten und dieses dann praktisch 'Verbalisieren': Z.B. o-negai suru, benkyoo suru. [Sel96](p.43) bezeichnet diese als 'periphrastic'. Auch hier gibt es Vollformen und Stämme.

- Andere Verben

  Vollformen in den Transkriptionen enden auf *-ru* (bzw.*-u*), wie zum Beispiel *toreru, kakaru, aru*. Ihr Lexikoneintrag enthält die jeweiligen Subkategorisierungs-Features und die Head-Features, wobei bei diesen Formen $tense = present$ und $form = plain$ gilt. Verbstämme enthalten nur Subkategorisierungsinformation, da die Head-Information von den Endungen kommt.

- Flexionsendungen

  Flexionsendungen liefern die Head-Information. Finite Endungen sind *-masu*, *-ta*, *-ru*, *-teru*, *-iru*, *-yoo* und *-nai*. Nicht finite Endungen sind *-te*, *-ba*, *-tara*, und *-tai*. Die Endung *-te* liefert keine Tempus-Information. Als Präsens-Endungen stehen *-masu*, *-ru*, *-ba*, *-tara*, *-tai*, *-iru* und *-nai* zur Verfügung. Past-Endung ist *-ta*. *-teru* hat das Feature $tense = prog$. *-yoo* ist eine Futur-Endung. *-te*, *-masu*, *-ta*, *-ru*, *-teru*, *-iru* und *-yoo* sind Indikativ-Endungen. *-tai* hat den Modus Voluntativ, *-nai* hat den Modus Negativ. Honorifische Formen sind *-masu*, *-ba*, *-tara* und *-yoo*, einfache Formen sind *-ru*, *-teru*, *-tai*, *-iru* und *-nai*. *-ta* hat keine honorifische Information, da diese in Ausdrücken wie *wakari mashi ta* aus dem Infix *mashi* kommt. Die Flexionsendung *n* ohne Features wird z.B. für *ari mase N* gebraucht (Eigenart der Segmentierung).

  Infixe (Eigenart der Segmentierung) liefern honorifische Information (*-mashi, -mase*) oder Modus-Information (*-mase*).

6 Die Grammatik

6.1 PP

Alle Auftreten von nominalen Phrasen mit einer Postposition werden als PP bezeichnet. Dies geschieht aus der grundlegenden Annahme heraus, daß in solchen Phrasen die Postposition als Head angesehen werden sollte. Die syntaktische Information, die die Postposition liefert, ist diejenige, die für die Subkategorisierung gebraucht wird. Dabei wird zwischen PPs mit Kasuspartikeln und anderen PPs (z.B. mit Genitivpartikel oder *to*) kein kategorialer Unterschied gemacht.

Das Makro hfp_p gibt die SYN-Features vom Head an den Mutterknoten weiter. Eine weitere Regel steht für Kombinationen von Postpositionen, wie z.B. *de mo, de wa, kara mo*.

In der gesprochenen japanischen Sprache gibt es das Problem der fehlenden Postpositionen. Dafür gibt es einige Sonderregeln, die so formuliert werden sollen, daß es nicht zu einer Lesartenexplosion kommt. Der Titel kann betroffen sein:

sensee	ogenki	desu	ka
Professor	gesund	COP	QUE

Herr Professor, geht es Ihnen gut?

Ebenso Pronomina

watakushi	harada	desu	kedomo
ich	Harada	COP	KONJ

Ich bin Harada

Zeitausdrücke

soredewa	juu	saN	nichi	no	gozeNchuu	sochira	no
KONJ	10	3	Tag	GEN	Vormittag	Sie	GEN

hoo	ni	ukagawa	se	te	itadaku
Seite	NI	GEHEN	HON	TE	HON

to	yuu	koto	de	ikaga	deshoo
TO	YUU	NOM	DE	recht	COP-QUE

Ist es Ihnen dann recht, wenn ich am 13. vormittags bei Ihnen vorbeikomme?

Ausdrücke mit Determinatoren

hai	toku	ni	atakushi	kono	hi	yotee
ja	besonders	NI	ich	jener	Tag	Plan

ga	ire	te	ori	mase	N	node
GA	eingefügt	TE	HON	NEG		weil

ja, an jenem Tag gibt es keinen besonderen Termin

NPs mit Konjunktionen

soo	desu	ne	dewa	jikkeN	to	mata	sono
so	COP	NE	dann	Experiment	und	dann	diese

juNbi	iroiro	jikaN	ga	kakaru	to
Vorbereitung	einige	Zeit	GA	brauchen	TO

omoi	masu	node
denken	HON	KONJ

hmm, dann brauchen das Experiment und auch seine Vorbereitung einige Zeit, denke ich

Eine speziell definierte Gruppe von Nomen, eventuell kombiniert mit Adjektiven, honorifischen Präfixen und Genitiv-PPs

hoNjitsu	wa	go	tsugoo	ikaga	deshoo	ka
heute	WA	HON	Umstände	recht	COP	QUE

wäre es Ihnen heute recht?

juu	nana	nichi	wa	watakushi	no	hoo	wa
10	7	Tag	WA	ich	NO	Seite	WA

toku	ni	nani	mo	yotee	wa	ima	no
besonders	NI	etwas	irgendein	Termin	WA	jetzt	NO

tokoro	hait	te	ori	mase	N	node
Zeitpunkt	eingefügt	TE	HON	NEG		KONJ

jikaN	aru	to	omoi	masu
Zeit	haben	TO	denken	

Am 17. gibt es bisher auf unserer Seite keinen Termin, ich denke, dass ich Zeit habe

Diese PPs werden als topikalisiert bezeichnet.

Eine spezielle Art der Topikalisierung in der gesprochenen japanischen Sprache ist, die Postposition *wa* durch *desu ne* zu ergänzen (*jitsu wa desu ne, kyoo wa desu ne*, Wahrheit-Topic, heute-Topic).

So wie Adjektive können sich die Konstruktionen *PP kara PP made* und *PP kara* verhalten:

juu	ji	kara	saN	ji	goro	made	desu	ne
10	Uhr	von	3	Uhr	etwa	bis	COP	NE

Das ist von 10 Uhr bis etwa 3 Uhr, nicht wahr?

ichi	ji	sugi	kara	deshi	tara
1	Uhr	nach	von	COP	KONDITIONAL

wenn das nach ein Uhr wäre

Eingebettete Sätze werden mit einer Postposition *to* oder einer Topikpartikel markiert.

juu	nana	nichi	de	o	negai	shi	tai
10	7	Tag	DE	HON	Bitte	tun	wollen

to	omoi	masu
TO	denken	HON

ich würde Sie gern um den 17. bitten

Sie können auch mit einer Nominalisierung auftreten:

kyoo	o	deNwa	sase	te	itadaki	mashi	ta
heute	HON	Telefon	tun	TE	HON	HON	PAST

no	wa
NOM	TOPIC

Die Angelegenheit, wegen der ich heute angerufen habe

Schließlich gibt es leere PPs, bei deren Auftreten in den Subkategorisierungsrahmen des Verbs *empty* eingetragen wird (sonst *real*). Im Baum steht 'ZPRO'. Diese Möglichkeit besteht für Nullpronomina, die im Japanischen besonders häufig in der gesprochenen Sprache auftreten (vgl. [MS94], [SM94]).

6.2 NP

Bei der Formulierung von Regeln für Nominalphrasen werden zunächst Besonderheiten der Domäne berücksichtigt. Datums- und Uhrzeitenangaben haben ebenso wie Personennamen und Institutionsnamen ihre eigene Syntax. Zeitausdrücke erfordern eine eigene Behandlung, daher gibt es die Syntaxregel NP $\rightarrow$ ZEIT. Ein Zeitausdruck kann aus einer Datums- oder Tagesangabe bestehen (*kayoobi*, Freitag), es kann Verknüpfungen zwischen Zeitangaben geben (*juu ichi nichi mokuyoobi*, 10 1 Tag Donnerstag), er kann aus Zahlen und numeralen Klassifikatoren bestehen (*juu ichi nichi, juu ji*, 10 1 Tag, 10 Uhr), oder aus Uhrzeitenangaben (*ichi ji han*, 1 Uhr halb). Personennamen können aus Name und Titel (*satoo sensei*, Satoo Prof.), nur dem Titel (*sensee*, Prof.), nur dem Namen (*harada*) oder Vor- und Nachnamen (*harada hitoshi*) bestehen. Ein Institutionsname besteht aus einem Namen und der Institutionsart (*okayama keNkyuusho*, Okayama-Institut).

Es gibt die Möglichkeit, der NP eine attributive PP, die mit *no* markiert ist, voranzustellen. Beispiele dafür sind:

a)	*satou*	*sensei*	*no*	*sukejuuru*
	PN	*Prof.*	*GEN*	*Terminplan*

b)	*go-tsugou*	*no*	*yoroshii*	*toki*
	HON-Umstände	*GEN*	*günstig*	*Zeit*

c)	*kyodokenkyu*	*no*	*tame*	*no*	*uchiawase*
	gemeinsame Forschung	*GEN*	*wegen*	*GEN*	*Treffen*

d)	*hatsuka*	*no*	*gogo*
	20.	*GEN*	*Nachmittag*

Um Verschachtelungen wie im Beispiel c) verarbeiten zu können, ist die Syntaxregel rekursiv definiert:

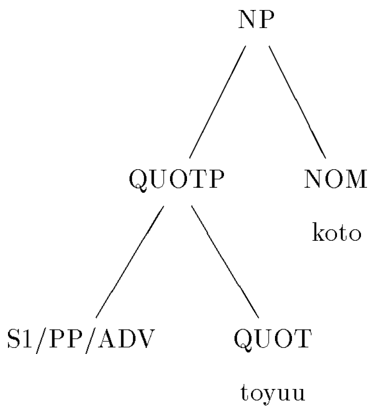
$$NP \rightarrow PP(gen) \quad NP$$

Dies ist aber natürlich eine Möglichkeit für Ambiguitäten, die semantisch gelöst werden müssen.

Eine Besonderheit der japanischen Nominalphrasen sind solche, die eine Nominalisierung enthalten. In den Dialogdaten gibt es ein Beispiel dafür:

hatsuka	no	gogo	kara	toyuu	no
20.	GEN	Nachmittag	von	QUOT	NOM

Eine NP kann daher aus einer Phrase mit *toyuu* (hier: QUOTP) und einer Nominalisierung oder einem Nomen bestehen. Die Phrase QUOTP ist in diesem Fall eine PP mit der Quotierung *'toyuu'*, sie kann aber auch zum Beispiel einen Satz oder ein Adverb enthalten:



Weiterhin gibt es Konjunktionen in der NP

juu	ni	nichi	to	juu	saN	nichi	deshi	tara
10	2	Tag	und	10	3	Tag	sein	Konditional

Determinativa (*kono hi*, dieser Tag) oder Adjektive (*o isogashii tokoro, iroiro go muri*, HON beschäftigt Zeit, verschiedenes HON überflüssiges). Außerdem kann eine NP nominalisiert werden

gogo	no	ichi	ji	made	to	yuu	koto	na	N
Nachmittag	NO	1	Uhr	bis	TO	YUU	NOM	NA	NOM

Nomen können auch ein honorifisches Präfix enthalten (*o kaigi*, HON-Treffen) oder zusammengesetzt sein (*kokusai kaigi*, internationales Treffen).

6.3 VP

Grundsätzlich besteht eine VP aus einer PP und einer untergeordneten VP, TVP genannt. Die PP wird als Subjekt, Objekt oder Objekt(2) oder als leeres Objekt oder Objekt(2) eingeordnet. Die Möglichkeit, im Japanischen die Satzargumente zu vertauschen (Scrambling) oder auch wegzulassen (Nullpronomina) wird dadurch realisiert, daß ein Verbargument nicht durch die Phrasenstruktur, sondern durch seine Markierung mit einer Postposition im Zusammenhang mit der Verbvalenz identifiziert wird.

Bei intransitiven Prädikaten kann die VP aus einer Adverbialphrase und einem Verb bestehen:

seminaa	ga	ichi	nichi	juu	hait	te	ori	masu
Seminar	GA	1	Tag	lang	eingefügt	TE	HON	HON

Sie kann auch nur aus einem Verb (*wakari mashi ta*, verstehen-Past) bestehen. Im Fall von Kopula-Prädikaten besteht die VP aus einer NP und der Kopula oder einem Adjektiv und der Kopula (*shiomi desu, yoroshii desu*, Shiomi COP, recht COP). Zwischen Adjektiv und Kopula kann eine Nominalisierung stehen (*ii no desu*, gut NOM sein). Das Argument wird mit dem Makro *x_comp* als X-Komplement ohne Kasusmarkierung eingeordnet. Vor allem in der gesprochenen Sprache kann die VP auch nur aus einem Adjektiv bestehen:

toriaezu	heejitsu	ga	ii	keredomo
zunächst	Werktag	GA	gut	KONJ

Sie wird dann als nicht finit, nicht honorifisch und indikativ mit einem subkategorisierten Subjekt bezeichnet.

Die TVP kann wiederum aus einer PP und einem Verb bestehen, wobei die PP Adjunkt ist:

kokusai	kaigi	o	raigetsu	ni	yotee
Internationales	Treffen	WO	nächsten Monat	NI	Plan
shi	masu				
tun	HON				

Sie kann aber auch einfach aus einem Verb bestehen:

yotee	wa	tore	masu
Zeit	TOPIC	nehmen können	HON

Es wurde eine V1-Kategorie eingeführt, um komplexe Verbformen behandeln zu können, die häufig eine honorifische Funktion haben. Dazu gehören Komplexe aus Nomen und Verben der 'suru-Klasse' (*keNkyuushitsu ni o ukagai shi masu*, Forschungsinstitut NI HON besuchen tun). Eine Aussage kann abgeschwächt werden, indem hinter dem Verb eine Nominalisierung und ein Kopula stehen (*yotee ga toreru* N *desu ga*, Zeit GA nehmen können NOM COP SAP). Der Verbkomplex kann durch ein Kopula erweitert werden (*ari mase N deshi ta, ukagawasashi te itadaki tai desu, ai te rare masu deshoo*, sein-NEG COP, aufsuchen-HON COP, frei sein können COP) Ein Verbkomplex kann weiterhin ein Auxiliar enthalten:

o	deNwa	sase	te	itadaki	mashi	ta
HON	Telefon	tun	TE	HON	HON	PAST

6.4 Verbmorphologie

Da VERBMOBIL kein eigenes Modul für die Analyse der japanischen Morphologie vorsieht, ist es Aufgabe der Syntax, diese zu integrieren. Die Formulierung der Verbmorphologie in der Syntax ist stark von der Segmentierung in den Transkriptionen abhängig, da diese das Trainingsmaterial für den Spracherkenner bilden. Ein Verb (V) kann aus einer verbalen Vollform (*toreru*) oder einem Verbstamm und einer oder mehrerer Verbendungen (*tore tai*) bestehen. Die Beziehung zwischen Stamm und Endungen ist durch das Makro *morph* ausgedrückt[4]. In diesem werden die Features des Stamms und der Endung(en) miteinander unifiziert und an den Mutterknoten weitergegeben. Für das Verb *omoimasu* kommt beispielsweise die Information über Subkategorisierungen vom Verbstamm *omoi* und die Information über Finitheit von der Verbendung *masu*. Verbendungen können -bedingt durch die Segmentierung - ein Infix enthalten (*tabe tai, tabe te ru, ita dake reba*). Für Auxiliare und Verben aus der Suru-Kategorie gilt das Gleiche. Ein honorifisches Merkmal ist die Kombination eines Verbs in der Te-Form mit einem honorifischen Auxiliar. Dabei wird die Head-Information (vor allem Tempus) vom Auxiliar übernommen und die Subkategorisierungsinformation vom Hauptverb. Dabei bestehen auch Auxiliare wiederum aus Verbstamm und Verbendung, wobei der Verbstamm keine Subkategorisierungsinformation enthält. Auxiliare können von Nominalisierung und Kopula gefolgt sein:

o	deNwa	kakesase	te	itadai	teru	N	desu
HON	Telefon	anrufen	TE	HON	TE	NOM	COP

6.5 Sätze

Die Grundregel für Sätze lautet S $\rightarrow$ PP VP. Die PP ist dabei entweder Subjekt, Objekt oder Objekt(2). Dies wird durch das Makro *complement* realisiert [5].

[4] Siehe Anhang Makros
[5] Siehe Anhang Makros

complement prüft, ob die PP nicht leer ist, sucht dann eine Kasusübereinstimmung der PP mit einem Argument im Subkategorisierungsrahmen des Verbs und vermerkt das gefundene Argument im Subkategorisierungsrahmen als $found = yes$. Die syntaktische Information der PP wird unter *match* eingetragen. Ist die PP leer, so wird sie mit dem Makro *empty_complement* als leeres Argument eingetragen. Dazu wird geprüft, ob das Argument noch nicht gefunden wurde und die Syntax des leeren Arguments wird in den Subkategorisierungsrahmen des Mutterknotens unter *match* unifiziert. Dabei handelt es sich vor allem um die Information, daß die PP leer ist (empty). Das Makro hfp_v sorgt dafür, daß die Head-Features und der Subkategorisierungsrahmen ohne das Feature *found* an den Mutterknoten weitergegeben werden.

Die Scrambling-Problematik wird dadurch (nichttransformationell) gelöst, daß durch das Makro *complement* eine PP durch ihre Postpositionsinformation im Zusammenhang mit der Sukategorisierungsinformation des Verbs identifiziert und in den Argumentrahmen des Verbs eingeordnet wird.

Vor dem S-Knoten können Adjunkte stehen. Dabei kann es sich um eine PP handeln, die Adjunkt oder Topic-Adjunkt ist:

roku	gatsu	no	yooka	wa	doo	desu	ka
6	Monat	NO	4.	WA	wie	COP	QUE

Wie ist der 4.6.?

chotto	yuugata	made	jugyoo	mo	hait	te	masu
etwas	Abend	bis	Termine	schon	eingefügt	HON	

Leider sind schon bis zum Abend Termine eingetragen

Außerdem gibt es die Möglichkeit, daß Adverbialphrasen wie in

doozo	yoroshiku	o	negai	itashi	masu
viel	Gruss	HON	Bitte	tun	HON

Herzlichen Dank

oder Interjektionen (*dewa, chotto, soredewa*) vor dem S-Knoten stehen. Ein Satz kann weiterhin auch einfach aus einem Idiom bestehen (*konnichiwa*, Guten Tag).

6.6 Äußerungen, Turns

Die Syntax unterscheidet Sätze (S), die die wichtigen syntaktischen Komponenten enthalten von Äußerungen (S1), die eine Referenz auf den Hörer enthalten (z.B. eine honorifische Form oder Flektion oder eine Satzendepartikel wie *keredomo* oder *ne*) und Turns, die aus einer Äußerung oder einer Sequenz von Äußerungen (S2) bestehen.

Ein Beispiel für eine Äußerung ist das folgende:

hiratsuka	desu	keredomo
Hiratsuka	COP	SAP

sensee	ogenki	desu	ka
Prof.	gesund	COP	QUE

Im Fall der Satzendepartikel bekommt der S1-Knoten die Finitheits-, Tempus- und Modusinformation vom S-Knoten. Die Äußerung wird als honorifisch eingestuft. Im Fall der Fragepartikel bekommt der S1-Knoten den Modus *que*; Finitheit, Tempus und Honorifizität werden vom S-Knoten übernommen. Ein Turn kann aus einer Äußerung bestehen, die allerdings ein honorifisches Merkmal im Prädikat haben sollte. Es werden auch Satzreihen akzeptiert, die mit oder ohne Satzkonjunktion verbunden sind. Bei fehlender Satzkonjunktion unterliegt die Verkettung allerdings Modus-Bedingungen. Wenn der Modus Indikativ, Potentialis oder Voluntativ ist, muß die erste Äußerung ein honorifisches Merkmal haben.

7 Schluß

Die japanische Syntax in VERBMOBIL besteht aus kontextfreien Phrasenstrukturregeln, die mit Features annotiert sind. Daher ist es möglich, eine transformationsfreie Syntax aufzubauen. Die Analyse gesprochener Sprache erfordert die Behandlung von Nullanaphern (als leere Knoten im Phrasenstrukturbaum), Satzendephrasen, die der Kommunikationssituation dienen und für die Anforderung geschriebener Sprache syntaktisch nicht wohlgeformte Phrasen (z.B. eine NP als Topik-Adjunkt).

Die Domäne hat einige syntaktische Besonderheiten. Sie erfordert eine Behandlung von Datumsangaben, Uhrzeitangaben, Personennamen mit Titel, sowie Institutionsnamen.

Literatur

[BMMM94] Johan Bos, Elsbeth Mastenbroek, Scott McGlashan, and Sebastian Millies. The Verbmobil semantic formalism (version 1.3. Verbmobil-Report 6, Universität des Saarlandes, 1994.

[BS92] U. Block and S. Schachtl. Trace & unification grammar. In *Proceedings of Coling*, 1992.

[BS94] U. Block and S. Schachtl. Compiling trace & unification grammar. In *Reversible Grammar in Natural Language Processing*, Kluwer Academic Publishers, 1994. Strzalkowski, T.

[CS94] R. Caspari and L. Schmidt. Parsing und Generierung in TrUG. Verbmobil-Report 40, München, 1994.

[Far84] A. Farmer. *Modularity in Syntax: A Study of Japanese and English.* Massachusetts: MIT Press, 1984.

[Gun87] T. Gunji. *Japanese Phrase Structure Grammar.* Dordrecht: Reidel, 1987.

[Kun73] S. Kuno. *The Structure of Japanese Language.* Cambridge: MIT Press, 1973.

[Miy86] S. Miyagawa. Predicate and numeral quantifier. In W.J. Poser, editor, *Papers from the Second International Workshop on Japanese Syntax*, pages 157–191. CSLI, 1986.

[MS94] D. Metzing and M. Siegel. Zero pronoun processing: Some requirements for a Verbmobil system. Verbmobil-Memo 46, Universität Bielefeld, 1994.

[Sel96] P. Sells. The projection of phrase structure and argument structure in Japanese. In T. Gunji, editor, *Studies in the Universality of Constraint-Based Structure Grammars*, pages 39–60. Osaka, 1996.

[SM94] M. Siegel and D. Metzing. Nullpronomina und die Organisation von Wissensquellen für den Transfer Japanisch-Englisch. Verbmobil-Memo 12, Universität Bielefeld, 1994.

A Grammatik

```
/******************** REGELN FUER PP'S ********************/

/****** GRUNDREGEL: ********/

pp ---> np , p |
        hfp_p(pp,p),
        pp:hon = np:hon,
        pp:wh=no,
         pp:tree= `pp(np:tree,p:tree).

pp ---> np_wh , p |
        hfp_p(pp,p),
        pp:hon = np_wh:hon,
        pp:wh = yes,
         pp:tree= `pp(np_wh:tree,p:tree).

/***********DOPPELTE PARTIKEL******/



pp ---> np , p:a , p:b  |
        (
        a:p_syn:p_head:case=de,
        b:p_syn:p_head:case in {no,mo};
        a:p_syn:p_head:case in {kara,made},
        b:p_syn:p_head:case in {ga,ni,top,mo}),
        hfp_p(pp,b),
        pp:hon = np:hon,
        pp:wh = no,
         pp:p_syn:p_head:empt=real,
         pp:tree= `pp(np:tree,a:tree,b:tree).



pp ---> np_wh , p:a , p:b  |
        (
        a:p_syn:p_head:case=de,
        b:p_syn:p_head:case in {no,mo};
        a:p_syn:p_head:case in {kara,made},
        b:p_syn:p_head:case in {ga,ni,top,mo}),
        hfp_p(pp,b),
        pp:hon = np_wh:hon,
        pp:wh = yes,
```

```
         pp:p_syn:p_head:empt=real,
         pp:tree= `pp(np_wh:tree,a:tree,b:tree).



/***************** FEHLENDE PARTIKEL *********/


pp ---> title |
        pp:p_syn:p_head:case=top,
        pp:p_syn:p_head:empt=real,
        pp:wh = no,
        pp:tree= `pp(title:tree).

pp ---> np_wh  |
        pp:p_syn:p_head:case=top,
        pp:p_syn:p_head:empt=real,
        pp:hon = np_wh:hon,
        pp:wh = yes,
         pp:tree= `pp(np_wh:tree).

pp ---> special_np |
        pp:p_syn:p_head:case=top,
        pp:p_syn:p_head:empt=real,
        pp:wh = no,
        pp:tree= `pp(special_np:tree).



special_np ---> special_nouns |
        special_np:tree = `special_np(special_nouns:tree).

special_np ---> adj , special_nouns |
        special_np:tree = `special_np(adj:tree,special_nouns:tree).

special_np ---> hon , special_nouns |
        special_np:tree = `special_np(hon:tree,special_nouns:tree).


special_np ---> pron |
        special_np:tree = `special_np(pron:tree).

special_np ---> zeit |
        special_np:tree = `special_np(zeit:tree).

special_np ---> pp , special_np:a |
        pp:p_syn:p_head:case=no,
```

```
        pp:p_syn:p_head:empt=real,
        special_np:tree = `special_np(pp:tree,a:tree).

pp --->  det , n |
         pp:p_syn:p_head:case=top,
        pp:p_syn:p_head:empt=real,
        pp:wh = no,
        pp:tree= `pp(det:tree,n:tree).


/************* TOPIKALISIERUNG MIT ...WA DESU NE *****/

pp ---> advp , p , cop , sap |
        p:p_syn:p_head:case=top,
        pp:p_syn:p_head:case=top,
        pp:p_syn:p_head:empt=real,
        pp:wh = no,
        pp:tree= `pp(advp:tree,p:tree,cop:tree,sap:tree).

pp ---> np , p , cop , sap |
        (p:p_syn:p_head:case=top;
         p:p_syn:p_head:case=made),
        pp:p_syn:p_head:case=top,
        pp:p_syn:p_head:empt=real,
         pp:wh = no,
       pp:tree= `pp(np:tree,p:tree,cop:tree,sap:tree).


/********* UNTERGEORDNETE SAETZE ***************/



pp ---> s , p  |
        hfp_p(pp,p),
        (p:p_syn:p_head:case=top;
         p:p_syn:p_head:case=to),
        pp:p_syn:p_head:empt=real,
        pp:wh = no,
        pp:tree= `pp(s:tree,p:tree).


pp ---> s , nom , p |
        s:v_syn:v_head:fin=yes,
        (s:v_syn:v_head:tense=past;
         s:v_syn:v_head:tense=pres),
        hfp_p(pp,p),
        (p:p_syn:p_head:case=top;
         p:p_syn:p_head:case=ni),
```

```
        pp:p_syn:p_head:empt=real,
        pp:wh = no,
        pp:tree= `pp(s:tree,nom:tree,p:tree).


/********************LEERE PP***********************/

pp ---> [] |
        pp:p_syn:p_head:empt=empty,
        pp:wh = no,
        pp:tree = `pp('ZPRO').


/******************* NP-REGELN ***************/

np ---> n1 |
        np:hon = n1:hon,
        np:tree = `np(n1:tree).

np ---> pn , title |
        np:tree = `np(pn:tree,title:tree).

np ---> title |
        np:tree= `np(title:tree).

np ---> pn , inst |
        np:tree = `np(pn:tree,inst:tree).

np ---> pn |
        np:tree= `np(pn:tree).

np ---> pn:a , pn:b |
        np:tree = `np(a:tree,b:tree).

/************* ZEITAUSDRUECKE ***************/

np ---> zeit |
        np:tree= `np(zeit:tree).

np_wh ---> zeit_wh |
        np_wh:tree= `np_wh(zeit_wh:tree).


/************* NO **************/

np ---> pp , np:a |
        pp:hon = a:hon,
        np:hon = a:hon,
```

```
        pp:p_syn:p_head:case=no,
        pp:p_syn:p_head:empt=real,


        np:tree = `np(pp:tree,a:tree).


/*************** TO YUU ********************/

np ---> quotp , nom |
        np:tree = `np(quotp:tree,nom:tree).


quotp ---> pp , quot1 |
         pp:p_syn:p_head:empt=real,
        (pp:p_syn:p_head:case=kara;
         pp:p_syn:p_head:case=made),
        quotp:tree = `quotp(pp:tree,quot1:tree).


quotp ---> s1 , quot1 |
        (s1:v_syn:v_head:modus=que;
         s1:v_syn:v_head:form=plain),
        quotp:tree = `quotp(s1:tree,quot1:tree).

quotp ---> adv , quot1 |
           quotp:tree = `quotp(adv:tree,quot1:tree).

quot1 ---> p , quot |
         p:p_syn:p_head:case=to,
          p:p_syn:p_head:empt=real,
         quot1:tree= `quot1(p:tree,quot:tree).

/************ NP MIT KONJUNKTIONEN ODER OHNE *******/

np ---> np:a , n_conj , np:b|
        np:tree= `np(a:tree,n_conj:tree,b:tree).


pp ---> np:a , n_conj:x , n_conj:y , np:b|
        pp:p_syn:p_head:case=top,
        pp:p_syn:p_head:empt=real,
        pp:wh = no,
        pp:tree= `pp(a:tree,x:tree,y:tree,b:tree).

/************ DEMONSTRATIVWOERTER KONO,SONO... ********/

np ---> det , n |
```

```
        np:tree= `np(det:tree,n:tree).

/*********** ADJEKTIVE IN DER NP ********/

np ---> adj , n1 |
        np:hon = n1:hon,
        np:tree = `np(adj:tree,n1:tree).

/************* NOMINALISIERUNGEN **********/

np ---> s , n1 |
        s:v_syn:v_head:fin=yes,
        s:v_syn:v_head:form=plain,
        np:tree= `np(s:tree,n1:tree).



np ---> np:a , na , nom |
        np:tree= `np(a:tree,na:tree,nom:tree).


/************* NOMEN ***********/
n1 ---> n |
        n1:tree = `n1(n:tree).


n1 ---> hon , n |
        n1:hon=yes,
        n1:tree = `n1(hon:tree,n:tree).


n1 ---> pron |
        n1:hon = pron:hon,
        n1:tree = `n1(pron:tree).

n1 ---> n:a , n:b |
        n1:tree = `n1(a:tree,b:tree).


/*********** ZEITAUSDRUECKE **********/

zeit ---> datum_tag |
        zeit:tree = `zeit(datum_tag:tree).

zeit_wh ---> datum_wh |
        zeit_wh:tree = `zeit_wh(datum_wh:tree).
```

```
zeit ---> zeit:a , zeit:b |
        zeit:tree = `zeit(a:tree,b:tree).

zeit ---> cardinal_p , numcl |
        zeit:tree = `zeit(cardinal_p:tree,numcl:tree).

/* NEU: VORINTEGRATION 2, um ku nan nichi rauszufiltern*/

zeit_wh ---> card_wh ,  numcl |
        zeit_wh:tree = `zeit_wh(card_wh:tree,numcl:tree).

cardinal_p ---> cardinal |
        cardinal_p:tree = `cardinal_p(cardinal:tree).

cardinal_p ---> cardinal:a , cardinal:b |
        kit_subsumes(cardinal_p,cardinal),
        cardinal_p:tree = `cardinal_p(a:tree,b:tree).

cardinal_p ---> cardinal:a , cardinal:b , cardinal:c |
        cardinal_p:tree = `cardinal_p(a:tree,b:tree,c:tree).

zeit ---> zeit:a , temp_mod |
        zeit:tree = `zeit(a:tree, temp_mod:tree).

zeit_wh --->  zeit_wh:a , temp_mod |
        zeit_wh:tree = `zeit_wh(a:tree, temp_mod:tree).



/*********** ADJEKTIVE *************/

adj ---> i_adj |
        adj:tree= `adj(i_adj:tree).


adj ---> na_adj |
        adj:tree= `adj(na_adj:tree).


adj ---> hon , na_adj |
        adj:tree= `adj(hon:tree,na_adj:tree).


adj ---> adj_mod , na_adj |
        adj:tree = `adj(adj_mod:tree,na_adj:tree).

adj ---> hon , i_adj |
        adj:tree= `adj(hon:tree,i_adj:tree).
```

```
adj ---> pp:a , pp:b |
        a:p_syn:p_head:case=kara,
        a:p_syn:p_head:empt=real,
        b:p_syn:p_head:case=made,
        b:p_syn:p_head:empt=real,
        adj:tree = `adj(a:tree,b:tree).

/*************************** VP **************************/


vp ---> pp , tvp |
        (
         complement(vp,pp,tvp,obj);
          complement(vp,pp,tvp,obj2);
        empty_complement(vp,pp,tvp,obj);
        empty_complement(vp,pp,tvp,obj2)),
        hfp_v(vp,tvp),
        vp:v_syn:wh = pp:wh,
         vp:tree = `vp(pp:tree,tvp:tree).



vp ---> advp , v1 |
         v1:v_syn:sc:obj:case=no_arg,
        v1:v_syn:sc:obj2:case=no_arg,
        hfp_v(vp,v1),
        vp:v_syn:wh = no,
        vp:tree= `vp(advp:tree,v1:tree).


vp ---> v1 |
        v1:v_syn:sc:obj:case=no_arg,
        v1:v_syn:sc:obj2:case=no_arg,
        hfp_v(vp,v1),
        vp:v_syn:wh = no,
        vp:tree= `vp(v1:tree).

vp ---> pp , cop |
        hfp_v(vp,cop),
        pp:p_syn:p_head:case=kara,
        pp:p_syn:p_head:empt=real,
        x_comp(vp,pp,obj),
        vp:v_syn:wh = no,
        vp:tree= `vp(pp:tree,cop:tree).
```

```
vp ---> np , cop |
        hfp_v(vp,cop),
        x_comp(vp,np,obj),
        vp:v_syn:wh = no,
        vp:tree= ‘vp(np:tree,cop:tree).


vp ---> np_wh , cop |
        hfp_v(vp,cop),
        x_comp(vp,np_wh,obj),
        vp:v_syn:wh = yes,
        vp:tree= ‘vp(np_wh:tree,cop:tree).

vp ---> wh_adj , cop |
        hfp_v(vp,cop),
        x_comp(vp,wh_adj,obj),
        vp:v_syn:wh=yes,
        vp:tree= ‘vp(wh_adj:tree,cop:tree).



vp ---> adj , cop |
        hfp_v(vp,cop),
        vp:v_syn:wh=no,
        x_comp(vp,adj,obj),
        vp:tree= ‘vp(adj:tree,cop:tree).



vp ---> adj , nom, cop |
        hfp_v(vp,cop),
        vp:v_syn:wh=no,
        x_comp(vp,adj,obj),
        vp:tree= ‘vp(adj:tree,nom:tree,cop:tree).



tvp ---> pp , v1 |
        adjunct(pp,v1),
         hfp_v(tvp,v1),
        tvp:tree = ‘tvp(pp:tree,v1:tree).


tvp ---> v1 |
        hfp_v(tvp,v1),
       tvp:tree = ‘tvp(v1:tree).
```

```
v1 ---> v1:a , cop |
        v1:v_syn:v_head=cop:v_syn:v_head,
        v1:v_syn:sc=a:v_syn:sc,
        v1:tree= `v1(a:tree,cop:tree).



vp ---> adj  |
         vp:v_syn:v_head:fin=no,
        vp:v_syn:v_head:modus=ind,
        vp:v_syn:sc:subj:case=ga,
        vp:v_syn:sc:obj:case=no_arg,
         vp:v_syn:sc:obj2:case=no_arg,
        vp:v_syn:wh = no,
        vp:tree= `vp(adj:tree).

v1 ---> n1 , suru_v1 |
        hfp_v(v1,suru_v1),
        v1:tree= `v1(n1:tree,suru_v1:tree).



/************************* VERBMORPHOLOGIE ******************/

cop ---> cop_st , v_ends |
        morph(cop,cop_st,v_ends),
        cop:tree = `cop(cop_st:tree,v_ends:tree).

cop ---> cop_voll |
        hfp_v(cop,cop_voll),
        cop:tree = `cop(cop_voll:tree).

v1 ---> v |
        hfp_v(v1,v),
        v1:tree = `v1(v:tree).

v1 ---> v , nom , cop |
        v1:v_syn:sc=v:v_syn:sc,
        v1:v_syn:v_head:fin=cop:v_syn:v_head:fin,
        v1:v_syn:v_head:modus=v:v_syn:v_head:modus,
        v1:v_syn:v_head:tense= cop:v_syn:v_head:tense,
        v1:v_syn:v_head:form=cop:v_syn:v_head:form,
        v1:tree = `v1(v:tree,nom:tree,cop:tree).


v ---> v_stamm , v_ends |
        morph(v,v_stamm,v_ends),
```

```
        v:tree = ‘v(v_stamm:tree,v_ends:tree).

v ---> vollform |
        hfp_v(v,vollform),
        v:tree = ‘v(vollform:tree).

v_ends ---> vf |
        hfp_v(v_ends,vf),
        v_ends:tree = ‘v_ends(vf:tree).

v_ends ---> infix , vf |
        morph(v_ends,infix,vf),
         v_ends:tree = ‘v_ends(infix:tree,vf:tree).

v_ends ---> infix:a , infix:b , vf |
        morph(v_ends,b,vf),
         v_ends:tree = ‘v_ends(a:tree,b:tree,vf:tree).


suru_v1 ---> suru_v , v_ends |
        morph(suru_v1,suru_v,v_ends),
        suru_v1:tree = ‘suru_v1(suru_v:tree,v_ends:tree).

suru_v1 ---> suru_v |

        hfp_v(suru_v1,suru_v),
        suru_v1:tree = ‘suru_v1(suru_v:tree).

suru_v1 ---> suru_v1:a , nom , cop |
        suru_v1:v_syn:sc=a:v_syn:sc,
        suru_v1:v_syn:v_head:fin=cop:v_syn:v_head:fin,
        suru_v1:v_syn:v_head:modus=a:v_syn:v_head:modus,
        suru_v1:v_syn:v_head:tense= cop:v_syn:v_head:tense,
        suru_v1:v_syn:v_head:form=cop:v_syn:v_head:form,
        suru_v1:tree = ‘suru_v1(a:tree,nom:tree,cop:tree).


/******* AUXILIARE *******/

v1 ---> v1:a , aux |
        a:v_syn:v_head:fin=no,
        v1:v_syn:v_head = aux:v_syn:v_head,
        v1:v_syn:sc=a:v_syn:sc,
        v1:tree= ‘v1(a:tree,aux:tree).

aux ---> aux:a , nom , cop |
        aux:v_syn:v_head=cop:v_syn:v_head,
        aux:tree = ‘aux(a:tree,nom:tree,cop:tree).
```

```
aux ---> aux_stamm , v_ends |
        morph(aux,aux_stamm,v_ends),
        aux:tree = `aux(aux_stamm:tree,v_ends:tree).

aux ---> aux_voll  |
        hfp_v(aux,aux_voll),
        aux:tree = `aux(aux_voll:tree).


/************* ADVERBIALPHRASEN **************/

advp ---> adv |
        advp:tree= `advp(adv:tree).

advp ---> na_adj , p |
        (p:p_syn:p_head:case=ni;
         p:p_syn:p_head:case=de),
         advp:tree= `advp(na_adj:tree,p:tree).

advp ---> cardinal , numcl , adv |
        advp:tree = `advp(cardinal:tree,numcl:tree,adv:tree).

advp ---> adv , p |
         (p:p_syn:p_head:case=ni;
          p:p_syn:p_head:case=top),
         advp:tree= `advp(adv:tree,p:tree).


/***************************** S *************************/

s ---> pp , s:a |
         (adjunct(pp,a);
         topic_adjunct(pp)),
         hfp_v(s,a),
        (pp:wh = yes,
         s:v_syn:wh = yes;
         pp:wh = no,
         s:v_syn:wh = a:v_syn:wh),
        s:tree = `s(pp:tree,a:tree).

s ---> advp , s:a |
        hfp_v(s,a),
        s:v_syn:wh = a:v_syn:wh,
        kit_subsumes(advp,s),
        s:tree= `s(advp:tree,a:tree).
```

```
s ---> pp , vp |
        (complement(s,pp,vp,subj);
         complement(s,pp,vp,obj);
          complement(s,pp,vp,obj2);
        empty_complement(s,pp,vp,subj)),
        hfp_v(s,vp),
        (pp:wh = yes,
         s:v_syn:wh = yes;
         pp:wh = no,
         s:v_syn:wh = vp:v_syn:wh),
         s:tree = `s(pp:tree,vp:tree).


/******************* SATZENDEPARTIKEL *****************/

sapp ---> sap |
        sapp:tree= `sapp(sap:tree).

sapp ---> sap:a , sap:b |
        sapp:tree = `sapp(a:tree,b:tree).


/******************* Satzreihen ************************/

s1 ---> s , sapp |
        s1:v_syn:v_head:fin=s:v_syn:v_head:fin,
        s1:v_syn:v_head:modus=s:v_syn:v_head:modus,
        s1:v_syn:v_head:tense=s:v_syn:v_head:tense,
        s1:v_syn:v_head:form=hon,
        s1:v_syn:sc=s:v_syn:sc,
        s1:tree= `s1(s:tree,sapp:tree).

s1 ---> s , que |
        s1:v_syn:v_head:fin=s:v_syn:v_head:fin,
        (s:v_syn:wh=yes,
        s1:v_syn:v_head:modus=whq;
        s:v_syn:wh=no,
        s1:v_syn:v_head:modus=ynq),
        s1:v_syn:v_head:tense=s:v_syn:v_head:tense,
        s1:v_syn:v_head:form=s:v_syn:v_head:form,
        s1:v_syn:sc=s:v_syn:sc,
        s1:tree= `s1(s:tree,que:tree).

s1 ---> s |
         hfp_v(s1,s),
        s1:tree = `s1(s:tree).
```

B Makros

```
complement(M,D,H,FUNCT) short_for
        D:p_syn:p_head:empt=real,
        funct(D,FUNCT),
        H:v_syn:sc:FUNCT:found=no,
        D:p_syn:p_head:case=H:v_syn:sc:FUNCT:case,
        M:v_syn:sc:FUNCT:match=D:p_syn,
        M:v_syn:sc:FUNCT:found=yes.

empty_complement(M,D,H,FUNCT) short_for
        D:p_syn:p_head:empt=empty,
        funct(D,FUNCT),
        (H:v_syn:sc:FUNCT:found=no,
         H:v_syn:sc:FUNCT:case in {ga,o,to,ni},
         H:v_syn:sc:FUNCT:match=D:p_syn,
         M:v_syn:sc:FUNCT:found=yes).




x_comp(M,D,obj) short_for
        funct(D,obj),
        M:v_syn:sc:obj:match:p_head:case=no_case,
        M:v_syn:sc:obj:match:p_head:empt=real.


adjunct(D,H) short_for
        funct(D,adjunct),
         D:p_syn:p_head:empt=real,
        (D:p_syn:p_head:case in {de,e,kara,made,mo};
         D:p_syn:p_head:case=ni,
         H:v_syn:sc:obj2:case=no_arg).

topic_adjunct(D) short_for
        funct(D,top),
         D:p_syn:p_head:empt=real,
         D:p_syn:p_head:case=top.

morph(M,H1,H2) short_for
        M:v_syn=H1:v_syn,
        M:v_syn=H2:v_syn.


%BG 12.05.96
funct(Cat,Value) short_for
Cat:funct = Value.
```

```
hfp_p(M,H) short_for
        M:p_syn=H:p_syn.

hfp_v(M,H) short_for
        M:v_syn:v_head=H:v_syn:v_head,
        M:v_syn:sc:subj:case=H:v_syn:sc:subj:case,
        M:v_syn:sc:subj:match=H:v_syn:sc:subj:match,
         M:v_syn:sc:obj:case=H:v_syn:sc:obj:case,
        M:v_syn:sc:obj:match=H:v_syn:sc:obj:match,
         M:v_syn:sc:obj2:case=H:v_syn:sc:obj2:case,
        M:v_syn:sc:obj2:match=H:v_syn:sc:obj2:match.
```

C Deklarationen

```
input => f(tree,v_syn:v_syn,hpsg:hpsg).
input => f(tree,p_syn:p_syn,hpsg:hpsg).

pp => f(tree,funct:funct,p_syn:p_syn,hon:yesno,wh:yesno,hpsg:hpsg).
vp => f(tree,v_syn:v_syn,hpsg:hpsg).
tvp => f(tree,v_syn:v_syn,hpsg:hpsg).


s => f(tree,v_syn:v_syn,hpsg:hpsg).
s1 => f(tree,v_syn:v_syn,hpsg:hpsg).
s2 => f(tree,v_syn:v_syn,hpsg:hpsg).
utt => f(tree,v_syn:v_syn,hpsg:hpsg).


v => f(tree,v_syn:v_syn,hpsg:hpsg).

v_stamm => f(tree,v_syn:v_syn,hpsg:hpsg).
vf => f(tree,v_syn:v_syn,hpsg:hpsg).
v_ends => f(tree,v_syn:v_syn,hpsg:hpsg).
vollform => f(tree,v_syn:v_syn,hpsg:hpsg).
infix => f(tree,v_syn:v_syn,hpsg:hpsg).
v1 => f(tree,v_syn:v_syn,hpsg:hpsg).

suru_v => f(tree,v_syn:v_syn,hpsg:hpsg).
suru_v1 => f(tree,v_syn:v_syn,hpsg:hpsg).

aux =>  f(tree,v_syn:v_syn,hpsg:hpsg).
aux_voll =>  f(tree,v_syn:v_syn,hpsg:hpsg).
aux_stamm =>  f(tree,v_syn:v_syn,hpsg:hpsg).
```

```
cop => f(tree,v_syn:v_syn,hpsg:hpsg).
cop_st => f(tree,v_syn:v_syn,hpsg:hpsg).
cop_voll => f(tree,v_syn:v_syn,hpsg:hpsg).

np =>  f(tree,funct:funct,hon:yesno,hpsg:hpsg).
n =>  f(tree,hpsg:hpsg).
n1 =>  f(tree,hon:yesno,hpsg:hpsg).
nom =>  f(tree,hpsg:hpsg).
pn =>  f(tree,hpsg:hpsg).
pron =>  f(tree,hon:yesno,hpsg:hpsg).
title =>  f(tree,hpsg:hpsg).
inst =>  f(tree,hpsg:hpsg).
temp_mod => f(tree,hpsg:hpsg).
datum_tag => f(tree,hpsg:hpsg).
datum_wh => f(tree,hpsg:hpsg).
zeit => f(tree,hpsg:hpsg).
cardinal =>  f(tree,hpsg:hpsg).
cardinal_p =>  f(tree,hpsg:hpsg).
card_wh =>  f(tree,hpsg:hpsg).
zeit_wh =>  f(tree,hpsg:hpsg).
np_wh =>  f(tree,funct:funct,hon:yesno,hpsg:hpsg).
numcl =>  f(tree,hpsg:hpsg).
idiom => f(tree,hpsg:hpsg).
special_nouns =>  f(tree,hpsg:hpsg).
special_np =>  f(tree,hpsg:hpsg).

det => f(tree,hpsg:hpsg).
hon =>  f(tree,hpsg:hpsg).


na_adj =>  f(tree,hpsg:hpsg).
i_adj =>  f(tree,hpsg:hpsg).
wh_adj =>  f(tree,funct:funct,hpsg:hpsg).
na =>  f(tree,hpsg:hpsg).
adj_mod =>  f(tree,hpsg:hpsg).
adj =>  f(tree,funct:funct,hpsg:hpsg).

adv =>  f(tree,hpsg:hpsg).

advp =>  f(tree,hpsg:hpsg).

s_conj =>  f(tree,hpsg:hpsg).
n_conj =>  f(tree,hpsg:hpsg).

sapp =>  f(tree,hpsg:hpsg).
sap =>  f(tree,hpsg:hpsg).
interj =>  f(tree,hpsg:hpsg).
```

```
p => f(tree,funct:funct,p_syn:p_syn,hpsg:hpsg).


quotp => f(tree,hpsg:hpsg).
quot  => f(tree,hpsg:hpsg).
quot1  => f(tree,hpsg:hpsg).

que  => f(tree,hpsg:hpsg).



v_syn => f(v_head:v_head,sc:sc,wh:yesno).
p_syn => f(p_head:p_head).

p_head => f(case:case,empt:empt).
v_head => f(fin:yesno,tense:tense,modus:modus,form:form).

sc => f(subj:subj,obj:obj,obj2:obj2).


subj => f(case:case,match:p_syn,found:yesno).
obj => f(case:case,match:p_syn,found:yesno).
obj2 => f(case:case,match:p_syn,found:yesno).

case => {to,ga,no,de,o,ni,e,top,no_arg,kara,made,mo}.
yesno => {yes,no}.
empt => {real,empty}.

modus => {ind,cond,neg,que,imp,volunt,pot,consul,whq,ynq}.
form => {plain,hon}.
tense => {pres,past,no_tense,future,progr}.
```

D Beispieleinträge aus dem Lexikon

```
lexicon(sukejuuru,n) |
        n:tree= n(sukejuuru).

lexicon(hatsuka,datum_tag) |
        datum_tag:tree= datum_tag(hatsuka).

lexicon(roku,cardinal) |
        cardinal:tree= cardinal(roku).
```

```
lexicon(nichi,numcl) |
        numcl:tree= numcl(nichi).

lexicon(goro,temp_mod) |
        temp_mod:tree= temp_mod(goro).

lexicon(koto,nom) |
        nom:tree = nom(koto).

lexicon(tanaka,pn) |
        pn:tree = pn(tanaka).

lexicon(jikan,special_nouns) |
        special_nouns:tree= ‘special_nouns(jikan).

lexicon(kono,det) |
        det:tree= det(kono).

lexicon(ga,p) |
        p:p_syn:p_head:case=ga,
        p:p_syn:p_head:empt=real,
        p:tree=p(ga).

lexicon(wa,p) |
        p:p_syn:p_head:case=top,
        p:p_syn:p_head:empt=real,
        p:tree=p(wa).

lexicon(yo,sap) |
        sap:tree = sap(yo).

lexicon(ka,que) |
        que:tree = que(ka).

lexicon(yoroshii,i_adj) |
        i_adj:tree = i_adj(yoroshii).

lexicon(ikaga,na_adj) |
        na_adj:tree = na_adj(ikaga).

lexicon(o,hon) |
         hon:tree = hon(o).

lexicon(moo,adj_mod) |
        adj_mod:tree= adj_mod(moo).

lexicon(sorede,interj) |
        interj:tree = interj(sorede).
```

```
lexicon(mochiron,adv) |
        adv:tree = adv(mochiron).

lexicon(yuu,quot) |
        quot:tree = quot(yuu).

lexicon(to,s_conj) |
        s_conj:tree = s_conj(to).

lexicon(to,n_conj) |
        n_conj:tree = n_conj(to).

lexicon(konnichiwa,idiom) |
        idiom:tree = idiom(konnichiwa).

lexicon(wakari,v_stamm) |
        v_stamm:v_syn:sc:subj:case=ga,
         v_stamm:v_syn:sc:obj:case=o,
        v_stamm:v_syn:sc:obj2:case=no_arg,
        v_stamm:tree=v_stamm(wakari).


lexicon(ori,aux_stamm) |
        aux_stamm:tree=aux_stamm(ori).

lexicon(deshi,cop_st) |
        cop_st:v_syn:sc:subj:case=ga,
         cop_st:v_syn:sc:obj:case=no_case,
         cop_st:v_syn:sc:obj2:case=no_arg,
        cop_st:tree=cop_st(deshi).

lexicon(shi,suru_v) |
         suru_v:v_syn:sc:subj:case=ga,
         suru_v:v_syn:sc:obj:case=o,
         suru_v:v_syn:sc:obj2:case=no_arg,
         suru_v:tree=suru_v(shi).

lexicon(omou,vollform) |
        vollform:v_syn:sc:subj:case=ga,
        vollform:v_syn:sc:obj:case=to,
        vollform:v_syn:sc:obj2:case=no_arg,
        vollform:v_syn:v_head:fin=yes,
        vollform:v_syn:v_head:modus=ind,
        vollform:v_syn:v_head:tense= pres,
        vollform:v_syn:v_head:form=plain,
        vollform:tree=vollform(omou).
```

```
lexicon(mashi,infix) |
        infix:v_syn:v_head:form=hon,
        infix:tree=infix(mashi).

lexicon(ta,vf) |
        vf:v_syn:v_head:fin=yes,
        vf:v_syn:v_head:modus=ind,
        vf:v_syn:v_head:tense=past,
        vf:tree=vf(ta).
```