CZECH CLITICS IN HIGHER ORDER GRAMMAR[1]

Jirka Hana[2]

Abstract. This paper presents an analysis of certain aspects of Czech sentential clitics in Higher Order Grammar. I focus on the relative order of clitics within the clitic cluster. The overall aim of the paper is to show that constraints governing Czech sentential clitics, though quite complex, can be captured relatively easily within a higher order formalism such as Higher Order Grammar.

In this paper, I present an analysis of some aspects of Czech sentential clitics in Higher Order Grammar (C.f. Pollard and Hana, 2003) – a framework currently under development, based on higher order logic (Church, 1940). I start with a general discussion of clitic phenomena in Czech (§1 & §2); I then briefly and informally describe the HOG formalism and sketch the formalization of syntactic structures in HOG (§3). Finally, I discuss the treatment of several clitic-related issues in HOG (§4).

[1]The work on this paper was partly supported by the NSF grant 746 262.

[2]jirka dot hana at gmail dot com

First, a note about the presentation of examples: All clitics are given in italics with the clitics most relevant for a certain discussion usually in boldface. In glosses, morphological categories that are not systematically expressed in English are marked by subscripts. Cases exemplify this practice: N = nominative, A = accusative, D = dative, G = genitive, I = instrumental, L = locative. Moreover, these categories are marked only when relevant to a particular problem. The past-tense auxiliaries (forms of 'be') are glossed simply as *aux* together with the appropriate person and number (e.g., *jsem* – aux_{1sg}); the conditional auxiliaries are glossed as *would* (e.g., *byste* – $would_{2pl}$). To make reading easier for English speakers, prepositions like *of* for genitive or *to* for dative are added to the glosses of NPs, in addition to the case subscripts. Often, I use numerical subscripts to show the relation between clitics and the word governing them; the subscripts increase with the degree of embedding of the governors.

1. Clitics – What's so special about them?

Czech has exceptionally free word order in comparison with most other languages, especially English. Unlike English, where word order is mostly fixed and is mainly used to express grammatical functions, word order in Czech is used to express topic-focus structure (C.f. Sgall et al., 1986) and definiteness. Thus for example, the words in sentence (1) can be rearranged in all 24 possible ways. Each of the sentences has a different topic-focus structure, but all of them are grammatically correct.

(1) Včera Petr viděl Marii.
yesterday Peter$_N$ saw Mary$_A$
'Yesterday, Peter saw Mary.'

More precisely, Czech word order is very free as regards the possibility of moving entire phrases; virtually any scrambling is possible. However, scrambling resulting in discontinuous phrases is much less common.[3] It is limited to certain syntactic constructions and to sentences involving a so-called contrastive topic, as in (2). Examples of the two corresponding sentences without discontinuous phrases are given in (3) and (4):

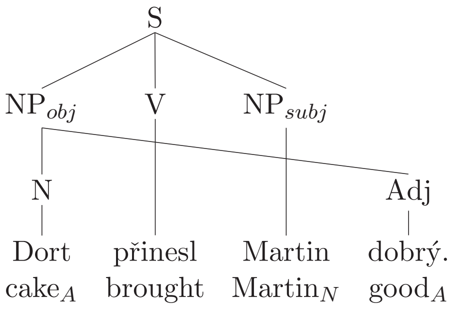
(2) a.

Dort	přinesl	Martin	dobrý.	(Ale	to	víno
cake$_A$	brought	Martin$_N$	good$_A$	(but	that	wine

bylo	bez	chuti.)
was	without	taste)

‘As far as the cake goes, Martin brought a good one. (But the wine was tasteless.)’

b.

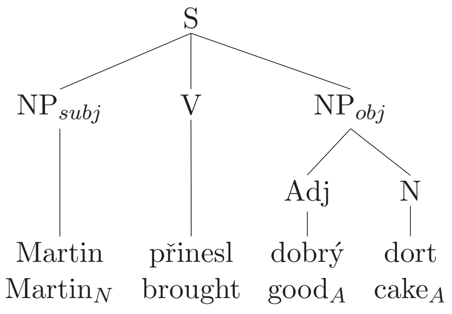


(3) a.

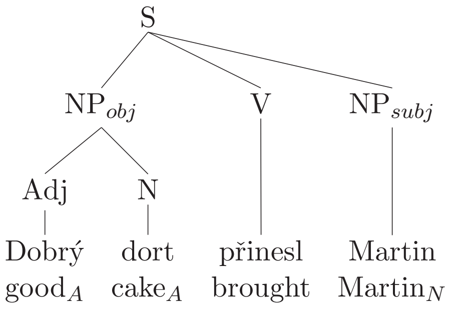
Martin	přinesl	dobrý	dort.
Martin$_N$	brought	good$_A$	cake$_A$

‘Martin brought a good cake.’

[3]Although it is far more common than in English. According to Holan et al. (1998), English allows a maximum of two discontinuity gaps in a phrase, while Czech does not impose any limit on the number of gaps. Of course, this is the competence point of view, the performance point of view is quite different – in a way parallel to, for example, relative-clause embedding which is also unlimited in competence but rather restricted in performance.

b.


(4) a. Dobrý dort přinesl Martin.
good$_A$ cake$_A$ brought Martin$_N$
'A good cake was brought by Martin'

b.


One phenomenon that is directly opposed to these two tendencies is sentential clitics. On one hand, their placement in sentences is very restricted – they occur most frequently in so-called Wackernagel or second position (Wackernagel, 1892) and even their ordering within this position is for the most part fixed. On the other hand, however, clitics frequently are associated with the presence of discontinuous phrases: while their position is restricted, the position of their potential governors (e.g., the verb subcategorizing for a clitic) is not.

The rigidity of clitic placement can be illustrated by comparing clitics to full NPs. The indirect object (*Petrovi* 'to-Peter$_D$') in sentence (5a) can also occur in any other place in that sentence (except within the PP) – for example in the topic position at the beginning of the sentence, as in (5b):

(5) a.

Dal	Petrovi	psa	k	vánocům.
gave	to-Peter$_D$	dog$_A$	for	Christmas

‘He gave Peter a dog for Christmas.’

b.

Petrovi	dal	psa	k	vánocům.
to-Peter$_D$	gave	dog$_A$	for	Christmas

‘To Peter, he gave a dog for Christmas.’

However, when the noun phrases here are replaced by the corresponding weak pronouns (one type of clitics), the above word-order freedom is lost – compare (5b) with the ungrammatical (6b):

(6) a.

Dal	*mu*	*ho*	k	vánocům.
gave	to-him$_D$	him$_A$	for	Christmas

‘He gave it to him for Christmas.’

b.

**Mu*	dal	*ho*	k	vánocům.
to-him$_D$	gave	him$_A$	for	Christmas

The clitics themselves have a fixed position within a clitic cluster. So, while the order of the direct object (*psa* ‘dog’) and the indirect object (*Petrovi* ‘to-Peter$_D$’) in sentence (5a) can be switched and still have the resulting sentence (7a) be fully grammatical, the corresponding change of word order in sentence (6a), with its clitics, results in the ungrammatical sentence (7b).

(7) a.

Dal	psa	Petrovi	k	vánocům.
gave	dog$_A$	to-Peter$_D$	for	Christmas

‘He gave Peter a dog for Christmas.’

b.

*Dal	*ho*	*mu*	k	vánocům.
gave	him$_A$	to-him$_D$	for	Christmas

The occurrence of multiple discontinuous phrases associated with clitics is also interesting. Sentence (8) is a normal sentence that can occur in everyday conversation. Yet the clitics *jsem*, *se*, *mu*, *to* here participate in several discontinuities, as the phrase structure in Figure 1 shows.

(8)

Opravit	*jsem*	*se*	*mu*	*to*	včera	snažil	marně.
to-repair	aux$_{1sg}$	refl$_A$	to-him$_D$	it$_A$	yesterday	tried	fruitlessly

‘I tried to repair it for him yesterday without success.’

In the rest of this paper, I discuss several aspects of clitic positioning in Czech within the framework of Higher Order Grammar. This is challenging for several reasons. First, Czech clitics are subject to constraints coming from many different linguistic levels: syntactic, morphological, phonological, pragmatic and stylistic. Second, some of the clitic regularities also differ from dialect to dialect or even from speaker to speaker. Third, it is hard to separate the relevant competence and performance constraints: is a particular sequence of clitics ill-formed because it is not grammatical or simply because it is too complicated from a performance point of view?

However, the goal of this paper is not to offer a precise partitioning of clitic data into well-formed and ill-formed sentences, but to show that a higher order formalism is suitable for describing such data, whatever may be the exact configurations of the parameters. I will focus especially on those aspects of Czech clitics that go beyond the capacity of context-free grammars. An area that is often difficult or cumbersome to describe for other grammar formalisms.

Another difficulty that arises in formalizing the grammar of Czech clitics – this time a problem of a completely different nature – stems from the fact that some of the constraints (e.g., related to clitic climbing) are more preferences

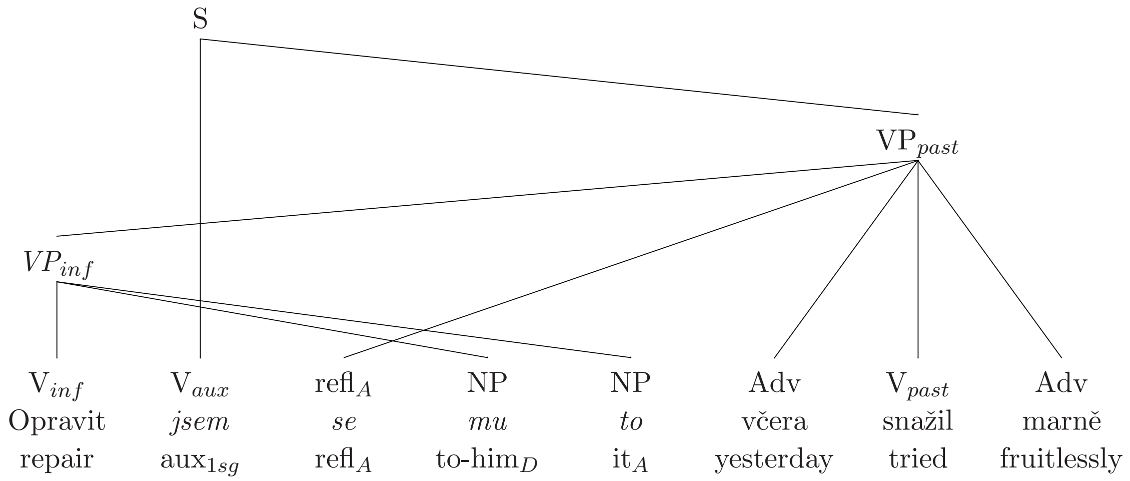


Figure 1: The syntactic structure of (8)

than they are strict rules. This is a major problem for symbolic (as opposed to statistical) linguistics, since none of the major formalisms is capable of capturing preferences. This conclusion holds for Higher Order Grammar as well, but limitations of time and space prevent me from discussing these issues in this paper.

2. Facts & Data

In the previous section, I introduced some of the properties of clitics in a informal way by comparing them with normal NPs. This section will discuss clitics in a more formal and structured way. First (§2.1), I show that Czech clitics cannot be defined on purely prosodic grounds, as traditional Czech grammars claim. Next (§2.2), I enumerate the most important clitics and their basic properties. The following subsection (§2.3) describes constraints on relative ordering of clitics based on their morpholexical properties. Finally (§2.4), I discuss various constraints on so-called clitic climbing. Some of the constraints are well known, but some, I believe, are original.

2.1. How can Czech clitics be characterized?

Traditionally, (e.g., Karlík et al., 1996:§44) Czech clitics are characterized as unstressed words which always form a prosodic word with the first stressed item in a clause. This is true in many sentences like (9) – the clitics *se mě* are unstressed and encliticize[4] with the first phrase in the clause *ten velký pes*. However, the general case is much more complicated. Below, I show that there are many cases where words that are consistently categorized as clitics do not encliticize, do bear stress, are preceded by more than one phrase, are preceded by less than one phrase, etc.

[4]Enclitics attach to the left, proclitics attach to the right; clitics is the general term.

(9) [Ten velký pes] ***se*** ***mě*** asi bojí.
that big dog refl_A of-me_G probably is-scared.
'That big dog is probably scared of me.'

2.1.1. Enclitics? Proclitics? Either? Neither?

Various papers show that Czech clitics are not always enclitics. Toman (1996) uses examples like (10) to show that Czech clitics do not always attach to the left, i.e., they are not always enclitics (| shows a prosodic boundary). In (10a), the clitic *se* forms a prosodic word with the material on its right, i.e., it procliticizes. It cannot encliticize, as (10b) shows. He argues that whether a clitic procliticizes or encliticizes is not a lexical property. The sentence in (11a) contains the same clitic *ji* twice: once as part of the subject governed by the infinitive *poslouchat*, and once governed by the verb *nudilo*. The prosodic boundary is identical with the syntactical boundary of the subject phrase, following *ji*. Therefore, in that case, *ji* encliticizes, while in the other it procliticizes.[5]

(10) a. Knihy, které tady vidíte, | ***se*** dnes platí
books which here see_{2pl} refl_A today pay
zlatem.
with-cold_I
"The books you can see here are paid for with gold today.

b. *Knihy, které tady vidíte, ***se*** | dnes platí zlatem.
(Toman, 1996)

(11) a. Poslouchat$_2$ *ji*$_2$, | *by*$_0$ *ji*$_1$ asi
to-listen her_A would_3 her_A probably
nudilo.
bore.

[5]The sentence contains two clitic clusters – the main one *by jí* and an embedded one containing just *jí*. Section 2.4 contains some information on multiple clitic clusters.

It would perhaps bore her (e.g., Ann) to listen to her (e.g., Mary).

b. *Poslouchat$_2$ | *ji*$_2$, *by*$_0$ *ji*$_1$ asi nudilo.

c. *Poslouchat$_2$ *ji*$_2$, *by*$_0$ | *ji*$_1$ asi nudilo.

d. *Poslouchat$_2$ *ji*$_2$, *by*$_0$ *ji*$_1$ | asi nudilo.

(Toman, 1996)

However, Oliva and Avgustinova (1995) show that clitics do not have to be a part of a larger prosodic unit at all. In the most natural pronunciation of (12), the prosodic boundaries both precede and follow the clitic *bychom* 'would$_{1pl}$'.

(12) My všichni, co spolu chodíme, | ***bychom***, | jak
we all, that together walk, would$_{1pl}$, as
říká Zilvar z chudobince, měli držet za
says Zilvar from poorhouse, should$_{pl}$ to-hold by
jeden provaz.
one rope

'As Zilvar from the poorhouse says, all of us friends should stick together.' (Oliva and Avgustinova, 1995)

Moreover, for some clitics, it is possible to bear stress and several clitics – e.g., *bychom* 'would$_{1pl}$', *bysme* 'colloquial would$_{1pl}$', *byste* 'would$_{2pl}$' – are bisyllabic.

One might argue that *bychom*, *bysme*, etc. are not clitics at all. However, regarding their placement in the sentence, they behave more as clitics than as normal verbs. They occur in Wackernagel's position, as other clitics do (13a, together with *se*), while normal verbs cannot (13b; the meaning analogous to 13a is in 13c). Unlike normal verbs, they also cannot occur outside the clitic cluster, e.g., topicalized at the beginning of a sentence (14a) or focused at the end of a sentence (15a). Also, unlike normal verbs, they cannot occur in isolation, e.g., as an answer to a question.

(13) a. Zítra *bychom* *se* viděli déle.
tomorrow would$_{1pl}$ refl$_A$ see longer
'Tomorrow, we would see each other for a longer time.'

b. *Zítra můžeme *se* vidět déle.
tomorrow can$_{1pl}$ refl$_A$ see longer

c. Zítra *se* můžeme vidět déle.
tomorrow refl$_A$ can$_{1pl}$ see longer
'Tomorrow, we can see each other for a longer time.'

(14) a. **Bychom* *se* viděli déle zítra.
would$_{1pl}$ refl$_A$ see longer tomorrow

b. Můžeme *se* vidět déle zítra.
can$_{1pl}$ refl$_A$ see longer tomorrow
'We can see each other for a longer time tomorrow.'

(15) a. *Zítra *se* viděli déle *bychom*.
tomorrow refl$_A$ see longer would$_{1pl}$

b. Zítra *se* vidět déle můžeme.
tomorrow refl$_A$ see longer can$_{1pl}$
'Tomorrow, we CAN see each other for a longer time.'

2.1.2. Following the first phrase?

While in most cases, clitics are preceded by a single top-level phrase, there are many exceptions. In (16), the clitic sequence *jsem se* 'aux$_{1pl}$ refl$_A$' follows only the noun part of the NP *levnou lednici* 'cheap$_A$ fridge$_A$'. On the other hand, in (17), again from Oliva and Avgustinova (1995), the clitic *se* 'refl$_A$' is preceded by three PP's.

(16) a. **Lednici** *jsme* *se*$_1$ nakonec rozhodli$_1$ koupit
fridge$_A$ aux$_{1pl}$ refl$_A$ finally decided to-buy
levnou.
cheap$_A$
'Regarding the fridge, we decide to buy a cheap one.'

b. **Levnou** *jsme* *se*$_1$ nakonec rozhodli$_1$ koupit
cheap$_A$ aux$_{1pl}$ refl$_A$ finally decided to-buy
lednici.
fridge$_A$
'It was a fridge, that we decided to buy cheaply.'

(17) [Od hrobky Caecilie Metelly na předměstí
from tomb of-Caecilia Metella on suburb
Říma] [přes vyprahlé roviny Apulie] [až po
of-Rome over dried plateaus of-Apulia up to
jižní pobřeží poloostrova] ***se***$_1$ jako nikde
southern coast of-peninsula refl$_A$ as never
nepřerušená rovná čára táhne$_1$ nejznámější ze
interrupted straight line runs most-famous from
všech antických cest – Via Appia.
all ancient roads – Via Appia.

'From the tomb of Caecilia Metella in the Rome suburbs over the dried plateaus of Apulia up to the southern coast of the peninsula runs the best known of all ancient roads, the Via Appia, in an uninterrupted straight line.'

(Oliva and Avgustinova, 1995)

2.1.3. Clitics in this paper

In this paper, I do not define Czech sentential clitics on purely prosodic grounds. Instead, I assume that Czech sentential clitics are words occupying Wackernagel's position.

This position can be identified using clear cases of prosodical clitics (e.g., weak pronouns), and then can be used to expand the set of clitics to the less obvious cases. Czech clitics are constrained especially by syntax and topic/focus structure, though prosody and other aspects of grammar also play some role. In the remainder of this paper, I focus on the ordering of clitics within Wackernagel's position and some of the rules that constrain clitic climbing; I will reserve issues connected with the formalization of the exact location of Wackernagel's position within sentences for a future paper.

2.2. The set of Czech clitics

Clitics are traditionally categorized as either constant or inconstant (see e.g., Karlík et al., 1996). Constant clitics always behave as clitics; inconstant clitics can function as clitics (can be in a clitic cluster) but can also function as normal words (that is they can occur outside of a clitic cluster and even bear focus).

Pronouns. It is traditional to distinguish weak and strong forms of pronouns. Weak forms (e.g., *ti* 'to-you$_D$') are constant clitics, strong forms (e.g., *tobě* 'to-you$_D$') are never clitics. Forms that can be either weak or strong (e.g., *nám* 'to-us$_D$') are inconstant clitics.

Reflexives. There are two simple reflexives – accusative *se* and dative *si*, and there are two contractions with the second-person singular of the past-tense auxiliary – accusative *ses* and dative *sis*. All of these are constant clitics.

Reflexives occur either as true reflexive pronouns (e.g., in *vidět se* – 'see oneself') or as particles with reflexive-tantum verbs (e.g., in *smát se* – 'laugh'). However, both these types of reflexive clitics seem to show exactly the same behavior.

		past	copula/passive	conditional
sg	1	jsem	jsem	bych/bysem
	2	jsi	jsi	bys/bysi
	3		je	by
pl	1	jsme	jsme	bychom/bysme
	2	jste	jste	byste
	3		jsou	by

Table 1: Auxiliary clitics

Auxiliary verbs. The verb *být* 'to be' can serve as an auxiliary or copula in Czech; the forms are summarized in Table 1. Some forms are constant clitics while others are inconstant. The forms of the past auxiliary[6] and the conditional auxiliary are constant clitics.[7] The non-negated passive auxiliary and copula forms are all inconstant clitics.

Však, prý, etc. The adverbs *však* 'however', *už* 'already', *prý* 'allegedly' and certain others are inconstant clitics.

-li. The case of the clitic *-li* is quite complicated. Oliva and Avgustinova (1995) argue that *-li* is not a sentential,

[6]Unlike in Serbo-Croatian, the third person of the Czech past tense is formed by a bare past participle without an auxiliary:

Czech:

(i) a. Psal jsem dopis.
wrote aux$_{1sg}$ letter$_A$
'I was writing a letter.'

b. Psal dopis.
wrote letter$_A$
'He was writing a letter.'

Serbo-Croatian:

(ii) a. Pisao sam pismo.
wrote aux$_{1sg}$ letter$_A$
'I was writing a letter.'

b. Pisao je pismo.
wrote aux$_{3sg}$ letter$_A$
'He was writing a letter.'

However, in passive, the auxiliary occurs in all three persons.

[7]The forms *bysem*, *bysi* and *bysme* are colloquial variants. The form *bysme* is closer to the official language than the other two forms.

but a word (lexical) clitic. Rosen (2001) claims that *-li* is at least sometimes a sentential clitic. I will leave this issue open, primarily because both cases can be very easily incorporated into the higher order formalization. The account below assumes that *-li* can be a sentential clitic.

Problematic cases. In addition, there are several problematic cases (e.g., *tu* 'here', *vlastně* 'actually', etc.) that are often (e.g., by Karlík et al., 1996:649–650) considered to be inconstant clitics, but Rosen (2001) convincingly argues that this is an illusion – the elements in question simply happen to occur next to a clitic cluster boundary.

2.3. Morpholexical ordering

As mentioned before, sentential clitics not only have a fixed position relative to the rest of the clause; they also have a quite fixed order relative to one another. A so-called clitic field can be quite complex: clitics from different verbs (or even adjectives, etc.) can cluster together in one place due to clitic climbing (see §2.4). In the present section, I describe a constraint which orders clitics based on morpholexical properties, so that certain clitics, and clitics in certain forms, must occur before certain other clitics. I present data and constraints that hold for Czech, but similar constraints are valid in other Slavic languages as well; for a comparison see, for example, Franks and King (2000).

The examples in (18) illustrate the basic point – the order of clitics in (18a) is grammatical, while the order in (18b) is not.

(18) a.

Martin	*se*	*jí*	*ho*	nakonec	rozhodl	koupit.
Martin$_N$	refl$_A$	to-her$_D$	him$_A$	finally	decided	to-buy

'Martin finally decided to buy it for her.'

b. *Martin *se* ***ho*** ***jí*** nakonec rozhodl
Martin$_N$ refl$_A$ him$_A$ to-her$_D$ finally decided
koupit.
to-buy

It is important to note that, for the relative acceptability of the sentences in (18), it is irrelevant whether or not the positioning of the verbs governing the relevant clitics (*rozhodl* 'decided' and *koupit* 'buy') yields more or less discontinuous phrases. Thus, consider the various possibilities in (19); the structure of (18a) is shown as (19a). The examples differ in their topic/focus structure, sometimes in very subtle ways, but all of them are grammatical.

(19) a. Martin *se*$_1$ *ji*$_2$ *ho*$_2$ nakonec rozhodl$_1$
Martin$_N$ refl$_A$ to-her$_D$ him$_A$ finally decided
koupit$_2$.
to-buy

'Martin finally decided to buy it for her.'

b. Martin *se*$_1$ *ji*$_2$ *ho*$_2$ koupit$_2$ nakonec
Martin$_N$ refl$_A$ to-her$_D$ him$_A$ to-buy finally
rozhodl$_1$. (Ale Eva ještě váhá)
decided

'Martin finally decided to buy it for her. (But Eva is still hesitating.)'

c. Koupit$_2$ *se*$_1$ *ji*$_2$ *ho*$_2$ nakonec rozhodl$_1$
to-buy refl$_A$ to-her$_D$ it$_A$ finally decided
Martin.
Martin

'It's Martin who finally decided to buy it for her.'

d. Rozhodl$_1$ *se*$_1$ *ji*$_2$ *ho*$_2$ nakonec koupit$_2$
decided refl$_A$ to-her$_D$ it$_A$ finally to-buy
Martin.
Martin

'It's Martin who finally decided to buy it for her.'

Basics. The examples in (18) and (19) show that reflexives (the accusative reflexive *se* and the dative reflexive *si*) precede nonreflexive dative pronouns (like *jí* 'to-her$_D$', *mi* 'to-me$_D$', etc.), which in turn precede nonreflexive accusative pronouns (such as *ho* 'him$_A$', similarly *mě* 'me$_A$', etc.). Schematically then:

(20) reflexives < dative < accusative[8]

Reflexives. The four relevant forms (accusative vs. dative and simple vs. contracted – see §2.2 above), are mutually exclusive. Only one of them can occur in a single clitic-cluster – this can be avoided by using non-clitic forms as in (21). For cases of reflexives governed by different heads see §2.4.1.

(21)	a.	*Smál	***se***	***si***.
		laughed	refl$_A$	refl$_D$

b.	Smál	***se***	(sám)	**sobě**.
	laughed	refl$_A$	(alone)	to-refl$_D$

'He laughed to himself.'

Datives. The situation with dative clitics is slightly more complicated, in that the ordering shown in (20) above holds only for complement dative clitics. There are two other types of nonreflexive dative clitics: ethical dative clitics and adjunct clitics. Second-person ethical dative clitics[9] can follow a reflexive as any other dative clitic, but they can also precede it, as both the examples in (22) and (23) show.

[8]Slovak, Slovenian and Sorbian follow the same pattern, but Serbo-Croatian requires reflexives to follow accusatives.

[9]As Rosen (2001) points out, in addition to the second person clitics *ti* 'to-you$_{SgD}$' and *vám* 'to-you$_{PlD}$', there is also a third-person plural ethical dative clitic *jim* 'to-them$_D$', formerly used in polite address. Such usage is now obsolete, and the second person plural pronoun is used instead.

Some speakers prefer them to precede the complement datives (23a, 23b), but many allow also the opposite order (23c):[10]

(22) a. On *se* ***ti*** vůbec nebál.
he refl$_A$ to-you$_D$ at-all not-scared
'You know, he wasn't scared at all.'

b. On ***ti*** *se* vůbec nebál.
he to-you$_D$ refl$_A$ at-all not-scared
'You know, he wasn't scared at all.'

(23) a. On *se* ***ti*** *jí* ani nepředstavil.
he refl$_A$ to-you$_D$ to-her$_D$ even not-introduced
'You know, he did not even introduce himself to her.'

b. On ***ti*** *se* *jí* ani nepředstavil.
he to-you$_D$ refl$_A$ to-her$_D$ even not-introduced
'You know, he did not even introduce himself to her.'

c. ?On *se* *jí* ***ti*** ani nepředstavil.
he refl$_A$ to-her$_D$ to-you$_D$ even not-introduced
'You know, he did not even introduce himself to her.'

The position of adjunct datives is after ethical datives/reflexives, as seen in (24), and before complement datives, as seen in (25):

[10]It appears that there is a great variety in speakers' constraints on the relative order of the ethical-dative clitics to the other dative clitics. However, all speakers perceive violations of their constraints on ethical dative placement as much less disturbing than violations of other constraints: e.g., violations of the relative ordering of dative and accusative clitics.

(24) a. Zbláznil *se* ***jí*** manžel.
Went-crazy refl$_A$ to-her$_D$ husband
'Her husband went crazy.' (Lit: The husband went crazy to her.)

b. *Zbláznil ***jí*** *se* manžel.
Went-crazy to-her$_D$ refl$_A$ husband

(25) a. On *se* ***mi*** *jí* ani nepředstavil.
He refl$_A$ to-me$_D$ to-her$_D$ even not-introduced
'He did not even introduce himself to her, for me.'
?'He did not even introduce himself to me, for her.'

b. On *se* ***jí*** *mi* ani nepředstavil.
He refl$_A$ to-her$_D$ to-me$_D$ even not-introduced
'He did not even introduce himself to me, for her.'
?'He did not even introduce himself to her, for me.'

Genitives. The example in (26) shows that genitive clitics follow accusative clitics:

(26) Napadá *mě* ***jich*** vždycky spousta.
come-upon me$_A$ of-them$_G$ always a-lot
'I always come upon a lot of them.'

Auxiliaries. As already explained in §2.2, some forms of the auxiliary verb *být* 'to be' (the past auxiliary, conditional auxiliary, non-negative passive auxiliary and non-negative copulas) are, or can be, clitics. They occur at the beginning of the clitic cluster.

(27) Martin ***by*** *se* *jí* *ho* nakonec rozhodl koupit.
Martin would$_3$ refl$_A$ to-her$_D$ him$_A$ finally decided to-buy
'Martin would decide to buy it for her at the end.'

-li. The questionable status of *-li* 'whether' as a sentential clitics was mentioned in §2.2. If we assume that it is a sentential clitic, then it is the first clitic in a clitic cluster, preceding even auxiliary verbs.

***však*, . . .** Such inconstant adverbial clitics as *však* 'however' and *prý* 'allegedly' can occur in virtually any position within a clitic cluster, as shown by the examples in (28) (probably all synonymous):

(28) a. Opravit ***však*** *jsem* *se* *mu* *to*
repair however aux$_{1sg}$ refl$_A$ to-him$_D$ it$_A$
včera snažil marně.
yesterday tried fruitlessly
'However, I tried to repair it yesterday without success.'

b. Opravit *jsem* ***však*** *se* *mu* *to* včera snažil marně.

c. Opravit *jsem* *se* ***však*** *mu* *to* včera snažil marně.

d. Opravit *jsem* *se* *mu* ***však*** *to* včera snažil marně.

e. Opravit *jsem* *se* *mu* *to* ***však*** včera snažil marně.

Since *však* is an inconstant clitic, it can also occur outside of a clitic clusters and can even be an occupant of the first position:

(29) **Však** *jsem* *se* *mu* *to* včera snažil
however aux$_{1sg}$ refl$_A$ to-him$_D$ it$_A$ yesterday tried
marně opravit.
fruitlessly repair
'However, I tried to repair it yesterday without success.'

For some speakers, *však* etc. are not full-fledged clitics. They allow them at a boundary of the clitic cluster (28a or 28e; whether it means that they are the first/last word in the cluster, or adjacent to the cluster), but they consider cases with clitic-cluster internal *však* (28b-28d) ungrammatical, or at least they do not prefer them.

However, even when we accept the examples in (28b-28d), it might be argued that *však* (and the other inconstant adverbial clitics) is in fact not a clitic at all. (a) It can be placed outside of the clitic cluster as in (29); and (b) it is not subject to the otherwise very strict clitic ordering (28). Under such an analysis *však* just breaks the clitic cluster into two parts. However, there is a question regarding the benefits of such an analysis and whether it would be worth the complications brought about by introducing the notion of "breakable" clitic clusters, moreover ones breakable by certain words but not by other ones. In addition, (a) holds for certain personal pronouns (e.g., *jí* 'she$_A$') and (b) are also true for ethical-dative clitics (see above). In the following, I assume that words like *však* can be part of the clitic cluster.

Summary of 2.3:

In Czech, clitics within a clitic cluster are ordered according to their morpholexical features:

(30) *-li* < aux < refl < adj. dat < compl. dat < acc < gen

In addition,

- ethical dative occurs anywhere after the position of auxiliaries and before the position of complement datives (or accusatives for some speakers),
- adverbial clitics like *však*, *prý*, or *už* can be freely positioned anywhere within a clitic segment. For some speakers, they can occur only at the beginning and end of the cluster.

2.4. Clitic Climbing

In a clause, clitics governed by the highest non-clitic governor (usually a non-auxiliary finite verb) obligatorily occur in Wackernagel position – the main clitic cluster. However, there can be other clitic clusters adjacent to more embedded words governing clitics. Clitics governed by those words can, or even tend, under certain circumstances occur in the less embedded clitic clusters, possibly in the main one. Within a finite clause, clitics governed by infinitives (31a), adjectives (31b), adverbs, and numerals (31c) can climb up into a higher clitic cluster. Note, that in most of the examples involving clitic climbing, I use numerical subscripts to show (a) which clitics belong to which governor (usually a verb), and (b) the degree of embedding of the governors.

(31) a.

Pomoct$_2$	najít$_3$	***by***$_0$	***se***$_1$	***mu***$_2$	***ho***$_3$
to-help	to-find	would$_3$	refl$_A$	to-him$_D$	him$_A$
určitě	snažil$_1$	i	Martin.		
definitely	tried	even	Martin		

‘Even Martin would try to help him to find it.’

b.

Marie	***si***$_4$	musí$_1$	začít$_2$	být$_3$	vědoma$_4$	svých
Marie	refl$_D$	must	to-start	to-be	aware	her
předností.						
strengths$_G$						

‘Mary must start to be aware of her strengths.’

[Rosen 2001:215]

c.

Martin	***jich***$_2$	ukradl$_1$	jen	pět$_2$.
Martin	of-them$_G$	stole	only	five

‘Martin stole only five of them.’

2.4.1. Well-known constraints

Clitic climbing is constrained by many rules, some of them stricter than others (see also Karlík et al., 1996; Rosen, 2001). In brief:

- A clitic cluster resulting from clitic climbing must be ordered according to the clitics' morpholexical features – recall §2.3.
- Clitics cannot climb out of certain types of phrases, namely, finite clauses, AdvPs, NPs with a verbal noun head, and normal AdjPs.
- A clitic cannot climb over another clitic governed by a less embedded category. In (32a), clitics stay with their verbs so that the only clitic in Wackernagel position is *se* 'refl$_A$'. In (32b), *mu* 'to-him$_D$' climbs from the verb *pomoci* 'to-help' to Wackernagel position, and *ho* 'him$_A$' climbs one level up, to the verb *pomoci.* Sentence (32d) is ill-formed, because *ho* 'him$_A$' has here climbed over *pomoci*, while the clitic *mu* 'to-him$_D$'(an argument of *pomoci*) does not climb. Of course, the surface ordering of verbs does not have to reflect their relative degree of embeddedness; c.f. (32e) & (32f).

(32) a. Všichni ***se***$_1$ snažili$_1$ ***mu***$_2$ pomoci$_2$ ***ho***$_3$ najít$_3$.
all refl$_A$ tried to-him$_D$ to-help him$_A$ to-find
'Everybody tried to help him to find it.'

b. Všichni ***se***$_1$ ***mu***$_2$ snažili$_1$ ***ho***$_3$ pomoci$_2$ najít$_3$.

c. Všichni ***se***$_1$ ***mu***$_2$ ***ho***$_3$ snažili$_1$ pomoci$_2$ najít$_3$.

d. *Všichni ***se***$_1$ ***ho***$_3$ snažili$_1$ ***mu***$_2$ pomoci$_2$ najít$_3$.

e. Pomoci$_2$ najít$_3$ ***se***$_1$ ***mu***$_2$ ***ho***$_3$ snažili$_1$ všichni.

f. Pomoci$_2$ ***mu***$_2$ ***ho***$_3$ najít$_3$ ***se***$_1$ snažili$_1$ všichni.

- Climbing cannot yield two phonologically identical clitics in a single clitic cluster. For example, in (11a), the first *jí* cannot climb to the main cluster and form *by jí jí* cluster. However, a clitic cluster can contain two identical clitics if they have the same governor.

- Climbing cannot result in several reflexive clitics in a single clitic-cluster (C.f. 2.3 for a similar constraint for reflexives with the same head). However, certain combinations of reflexive clitics can undergo haplology (See e.g., Rosen, 2001).

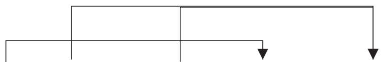
- Climbing is strongly preferred if the main verb is a modal verb (Karlík et al., 1996:651)

- For nonmodal verbs, climbing is preferred if its governor has no other dependents – dependents of the same governor tend to stay together.

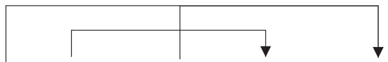
However, as the examples in the following sections will show, there is more going on here than just the above well-known constraints. The three constraints below are not completely orthogonal: the Continuous Clusters Constraint (CCC; §2.4.2) is predicted by the Ordering by Governors' Degree of Embeddedness Constraint (GDEC; §2.4.3) and there are cases that are explicable by both the GDEC and the Reflexives And Control Constraint (RCC; §2.4.4). However, it still makes sense to state them separately, because for some speakers the constraints are of different importance. As stated above, I do not attempt here to separate precisely the well-formed and ill-formed data; instead, I identify the types of constraints that a linguist would need in order to do this and show that those constraints are expressible in HOG.

2.4.2. The Continuous Clusters Constraint (CCC)

This constraint relates dominance and word order. Examples such as (33) suggest that, in a clitic cluster, all clitics of a common governor have to form a continuous sub-cluster. Of course, this holds only for clitics which are in the same clitic field: if a subset of the clitics climbs up to the clitic field but the rest stay down with a lower verb (which is grammatical but often not the most preferred option), then this constraint describes only the subset in the clitic field.



(33) a. Martin *se*$_1$ *jí*$_2$ *to*$_2$ rozhodl$_1$ koupit$_2$.
Martin refl$_A$ to-her$_D$ it$_A$ decided to-buy
'Martin decided to buy it for her.'



b. *Martin *se*$_2$ *jí*$_1$ *to*$_2$ přikázal$_1$ naučit$_2$.
Martin refl$_A$ to-her$_D$ it$_A$ ordered to-learn
intended: 'Martin ordered her to learn it.'

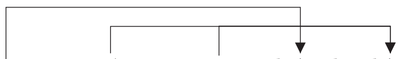
Sentence (33b) is not grammatical, and its intended meaning must be expressed without having all the clitics together – as, for example, in (34). The sentence in (33a) can be rephrased in a similar way, but most speakers prefer the variant with all the clitics in Wackernagel position. The constraint governing cases like these is expressed in (35).

(34) Martin *jí*$_1$ *přikázal*$_1$ *se*$_2$ *to*$_2$ naučit$_2$.
Martin to-her$_D$ ordered refl$_A$ it$_A$ to-learn
'Martin ordered her to learn it.'

(35) **Continuous Clusters Constraint (CCC)**:
All clitics in a clitic field with the same governor have to form a continuous cluster.

Sentence (33b) also violates the Reflexives and Control Constraint stated below (§2.4.4), however speakers feel about its incorrectness much more strongly than they do about the incorrectness of sentences that violate only the RCC without also violating the CCC.

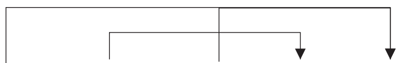
The CCC can even disambiguate sentences where several clitics have the same case:[11]



(36) a.

Martin	*mu*$_1$	*jí*$_2$	*to*$_2$	zakázal$_1$	dát$_2$.
Martin	to-him$_D$	to-her$_D$	it$_A$	forbid	to-give

‘Martin forbid him to give it to her.’



b.

*Martin	*mu*$_2$	*jí*$_1$	*to*$_2$	zakázal$_1$	dát$_2$.
Martin	to-him$_D$	to-her$_D$	it$_A$	forbid	to-give

‘Martin forbid her to give it to him.’

2.4.3. Ordering by Governors’ Degree of Embeddedness (GDEC)

Rosen (2001233) points out that in clitic clusters with multiple dative clitics, the latter have to be ordered according to the relative embedding of their governing verbs – a clitic governed by a more embedded verb follows a clitic with a less embedded verb.[12] In (37a), the dative pronoun *mu* ‘to-him$_D$’ goes before the dative pronoun *jí* ‘to-her$_D$’, therefore

[11]It might be argued that this constraint exists precisely to avoid such ambiguities. However, the majority of the sentences with clitics do not contain clitics in the same case and are therefore not ambiguous from this point of view. Still, the positions of these clitics are constrained by the CCC constraint. It seems odd that the relatively infrequent sentences involving clitics in the same case would be the reason for a constraint limiting clitic distribution in all the sentences involving clitic climbing. In any case, my goal is to *describe* the distribution of Czech clitics; I do not attempt to *explain* it.

[12]Rosen argues for this constraint on the bases of examples like (i). However, their acceptability can be independently explained by the

mu is governed by the highest verb – *zakázal* 'forbade' and *jí* by the embedded verb *kupovat* 'to-buy'. In (37a), the situation is reversed. Sentence (37c) shows that the linear order of the verbs is irrelevant, only their embedding is important.

(37) a. Martin ***mu***$_1$ ***jí***$_2$ včera zakázal$_1$
Martin to-him$_D$ to-her$_D$ yesterday forbade
kupovat$_2$ takové dárky.
to-buy such presents
'Martin forbade him to buy her such presents yesterday.'
?'Martin forbade her to buy him such presents yesterday.'

b. Martin ***jí***$_1$ ***mu***$_2$ včera zakázal$_1$
Martin to-her$_D$ to-him$_D$ yesterday forbade
kupovat$_2$ takové dárky.
to-buy such presents

Continuous Cluster Constraint (35). In (ia), the cluster *se mi* (clitics governed by *nepodařilo* 'not-succeeded') and the cluster *mu ho* (clitics governed by *poslat* 'to-send') are continuous. However, in (ib), the cluster *se ... mu* (*nepodařilo*) and the cluster *mi ... ho* (*poslat*) are discontinuous. The cases (ic) and (id) are analogous.

(i) a. Poslat$_2$ *se*$_1$ ***mi***$_1$ ***mu***$_2$ *ho*$_2$ dnes nepodařilo$_1$.
to-send refl$_A$ to-me$_D$ to-him$_D$ him$_A$ today not-succeeded
'I did not succeed in sending it to him today.'

b. ??Poslat$_2$ *se*$_1$ ***mi***$_2$ ***mu***$_1$ *ho*$_2$ dnes nepodařilo$_1$.
to-send refl$_A$ to-me$_D$ to-him$_D$ him$_A$ today not-succeeded
'He did not succeed in sending it to me today.'

c. Poslat$_2$ *se*$_1$ ***mu***$_1$ ***mi***$_2$ *ho*$_2$ dnes nepodařilo$_1$.
to-send refl$_A$ to-him$_D$ to-me$_D$ him$_A$ today not-succeeded
'He did not succeed in sending it to me today.'

d. ??Poslat$_2$ *se*$_1$ ***mu***$_2$ ***mi***$_1$ *ho*$_2$ dnes nepodařilo$_1$.
to-send refl$_A$ to-him$_D$ to-me$_D$ him$_A$ today not-succeeded
'I did not succeed in sending it to him today.'

‘Martin forbade her to buy him such presents yesterday.’
?‘Martin forbade him to buy her such presents yesterday.’

c.

Kupovat$_2$	takové	dárky	***mu***$_1$	***jí***$_2$
to-buy	such	presents	to-him$_D$	to-her$_D$
včera	Martin	zakázal$_1$.		
yesterday	Martin	forbade		

‘Martin forbade him to buy her such presents yesterday.’
?‘Martin forbade her to buy him such presents yesterday.’

It is necessary to note that many speakers tend to avoid having multiple datives in a single clitic cluster, especially with certain verbs. However, based on examples like (38), it seems that this constraint can be extended to any (constant?) clitics:

(38) a.

*Martin	*se*$_2$	*mu*$_1$	přikázal$_1$	usmívat$_2$.
Martin	refl$_A$	to-him$_D$	ordered	to-smile

intended: Martin ordered him to smile.’

b.

Martin	*se*$_1$	*mu*$_2$	odvážil$_1$	představit$_2$	Lucii.
Martin	refl$_A$	to-him$_D$	dared	introduce	Lucy

Martin dared to introduce Lucy to him.’

(39) **Ordering by Governors’ Degree of Embeddedness Constraint (GDEC)**
Clitics in a clitic field are ordered according to the degree of embedding of their governors: namely, a clitic governed by a less deeply embedded verb precedes a clitic governed by a more deeply embedded verb. The surface order of the governors is irrelevant.

2.4.4. Reflexives and Control Constraint

Consider the sentences without a climbing reflexive in (40), and the corresponding sentences with a climbing reflexive in (41).

(40) a. Martin zakázal$_1$ Petrovi dívat$_2$ ***se***$_2$ na televizi.
Martin forbid to-Peter$_D$ to-watch refl$_A$ on TV
'Martin forbid Peter to watch TV.'

b. Neviděl$_1$ *jsem*$_0$ ještě Martina mýt$_2$ ***si***$_2$ ruce.
not-seen aux$_{1sg}$ yet Martin$_A$ to-wash refl$_D$ hands$_A$
'I haven't seen Martin wash his hands yet.'

c. Vláda občanům doporučila$_1$ ***se***$_2$ pojistit$_2$.
government to-citizens$_D$ recommended refl$_A$ to-insure
'The government recommended the citizens get insurance.'

(41) a. ??Martin ***se***$_2$ zakázal$_1$ Petrovi dívat$_2$ na televizi.
Martin refl$_A$ forbid to-Peter$_D$ to-watch on TV
'Martin forbid Peter to watch TV.'

b. ??Neviděl$_1$ *jsem*$_0$ ***si***$_2$ ještě Martina mýt$_2$ ruce.
not-seen aux$_{1sg}$ refl$_D$ yet Martin$_A$ to-wash hands$_A$

‘I haven’t seen Martin wash his hands yet.’

c. ??Vláda ***se***$_2$ občanům doporučila$_1$
government refl$_A$ to-citizens$_D$ recommended
pojistit$_2$.
to-insure
‘The government recommended the citizens get insurance.’

All eight speakers who were consulted accepted the sentences without a climbing reflexive in (40), while most of them rejected the corresponding sentences with a climbing reflexive in (41). Some speakers did accept some of the sentence in (41), but they usually still preferred the corresponding sentences in (40). On the other hand, all questioned speakers accepted the sentences with a climbing reflexive in (42), and they preferred them over the corresponding sentences without climbing:

(42) a. Při výběru ***si***$_2$ zákazník musí$_1$
during selection refl$_D$ customer must
všímat$_2$ i spotřeby.
to-pay-attention also consumption$_G$
‘During selection, the customer must pay attention also to consumption.’

b. Ekonomika ***se***$_2$ začíná$_1$ zlepšovat$_2$.
economy refl$_A$ starts to-improve
‘The economy starts to improve.’

c. Martin ***se***$_2$ potřebuje$_1$ zeptat$_2$, jak …
Martin refl$_A$ needs to-ask how …
‘Martin needs to ask how … .’

d.

Martin	***se***$_2$	snažil$_1$	dokončit$_2$	všechno	včas.
Martin	refl$_A$	tried	to-finish	everything	on-time

‘Martin tried to finish everything on time.’

The difference between (41) and (42) involves the varying type of control of the embedded verb: the sentences in (41) involve object control, while the sentences in (42) involve subject control.

(43) **Reflexives and Control Constraint (RCC)**
Reflexive clitics do not (tend not to) climb from object-controlled VPs.

In fact, the 100K-sentence Prague Dependency Treebank (Bémova et al., 1999) does not contain a single sentence with a reflexive climbing from an object-controlled VP. However, it is necessary to stress that PDT is a newspaper corpus and clitics are much more common in spoken language than in written.

3. Higher Order Grammar

The higher order formalism used in this paper builds on Higher Order Grammar (HOG) as described by Pollard (2001b) and Pollard and Hana (2003), which in turn builds on higher order logic (HOL; Church, 1940). It is a framework still very much under development, and the version in this paper differs from the Pollard and Hana (2003) version in one very important aspect. Here the model of a linguistic expression is not an equivalence class of proofs but rather a functional structure encoding phrase structure and word order. That means, for example, that valency is not modeled by functional types, but is simulated using a principle similar to the subcategorization principle in Head-Driven Phrase Structure Grammar (HPSG; Pollard and Sag, 1994). That is a serious drawback (and gives rise to some of the problems mentioned in Pollard (2001a)), but, on the other hand, the focus on phrase structure keeps the present approach closer to mainstream linguistic views of language expressions. However, the two approaches are probably compatible, a possibility I am currently investigating.

3.1. HOG informally

For an overview of HOG syntax, see Appendix A. Here I will just informally mention the basic features of the formalism. HOG is based on typed higher order logic/lambda calculus (Church, 1940) and on Lambek's (1988, 1999) categorical (as opposed to categorial) grammar. The advantage of using HOL lies not only in its expressiveness, but also in the fact that it is a standard off-the-shelf formalism with extensive research already completed in both formal and computational areas.

Formally, the main difference between HOG and HPSG (Pollard and Sag, 1994), is that functions (and relations) in HOG are first class citizens; for example, they can be passed to other functions as arguments. In this respect, the formalism resembles a typed functional programming language like Haskell (`haskel.org`) or ML (Milner et al., 1997).

Constraints are nonlogical axioms and features (in the HPSG/LFG sense) are functions from one object to another. The type system consists of a set of basic types – like NP, S, etc. – and a set of type constructors. Type constructors include indexed cartesian products (for indexed tuples), coproducts and an exponential (for functional types). HOG has a very powerful kind of subtyping – for every type, and for every predicate on that type, there is a subtype whose members are those members of the supertype for which the predicate is true (this is written as $\lceil T \mid \phi \rceil$). Unlike HPSG types, HOG types are populated by terms (e.g., $\mathsf{Case} = \{\mathsf{nom}, \mathsf{acc}, \ldots\}$). Of course, for any term, it is possible to define a type populated just by that term ($\mathsf{Nom} = \lceil \mathsf{Case} \mid \mathsf{this} = \mathsf{nom} \rceil$).

The categorical model of HOG is a topos (and the logic is the internal language of a topos). Topos is a generalized set theory; see Lambek and Scott (1986) for details.

3.2. Why HOG? Why not HPSG?

A question one might ask is: Why not use HPSG? HPSG (Pollard and Sag, 1994) is one of the most commonly used syntactic formalisms, and probably the most frequently used formalism in computational linguistics. It has been successfully applied to many linguistic problems. However, formalization of grammars of languages with free-phrase order and relatively common discontinuous phrases, although theoretically possible, is complicated, nonmodular, and nonintuitive as Penn (1999) or Rosen (2001) show. Penn formulates his analysis of Serbo-Croatian clitics both in a precise natural-language and in HPSG. The contrast between the two analyses is striking – the former is elegant and simple, while the later is extremely complicated and cumbersome. A similar conclusion can be drawn from Rosen's formalization of Czech word order.[13]

HOG and HPSG have the same theoretical expressive power. However, HOG is based on higher order logic, while HPSG is based on RSRL (Relational Speciate Re-entrant Language; Richter, 2000). The former is a standard, well researched formalism, widely used in mathematics, computer science and other areas. The latter is a completely idiosyncratic formalism, unknown to anybody except a small group of formal linguists. For a more extensive discussion of HPSG problems, both formal and practical, see Pollard (2001a). Here, we limit ourselves to mentioning a very sur-

[13]Rosen does not use standard HPSG, but he uses RSRL (Richter, 2000), the underlying logic of HPSG.

prising fact that even finite models are undecidable in RSRL (Kepser, 2001)!

3.3. Grammar core

I assume that a sentence is a set of words plus two structures – a tectogrammatical tree and a topological tree. This roughly follows Curry (1961) (see also Sgall et al., 1969; Dowty, 1996; and others) in distinguishing tectogrammatics and phenogrammatics. Tectogrammatical structure captures argument structure. It is a flat phrase structure without word order: i.e., a dependency structure (in the Dependency Grammar sense), plus a grouping into phrases.

Topological trees are devices for constraining word order. They were introduced by Drach (1937) to describe German word order. In HPSG, they were first used by Reape (1994), and later elaborated by Kathol (1995). Penn (1999) generalized Kathol's approach, provided full formalization in RSRL and applied it to Serbo-Croatian clitics. Unlike Kathol's linearization account, Penn's topological trees are independent of the phrase structure. Rosen (2001) used Penn's approach to formalize Czech word order in Functional Generative Description (FGD; Sgall et al., 1986) with RSRL used as the underlying formalism.

A topological tree encodes a hierarchy of domain objects. Instead of providing formal definitions, I will show a self explanatory example of syntactic structures corresponding to sentence (44) – with tectogrammatical structure[14] in Figure 2 and topological tree in Figure 3.

[14]The linear order of the nodes is irrelevant. The convention is that a governor precedes its arguments and complements precedes adjuncts.

However, there may be other orderings of nodes relevant on the tecto-structure besides the phrase hierarchy. For example, Functional Generative Description (FGD; Sgall et al., 1986) captures topic-focus articulation/information structure as an ordering of the nodes in the tecto-structure.

(44)

Opravit	*jsem*	*se*	*mu*	*to*	včera	snažil	marně.
repair	aux$_{1sg}$	refl$_A$	to-him$_D$	it$_A$	yesterday	tried	fruitlessly

‘I tried to repair it for him yesterday without success.’

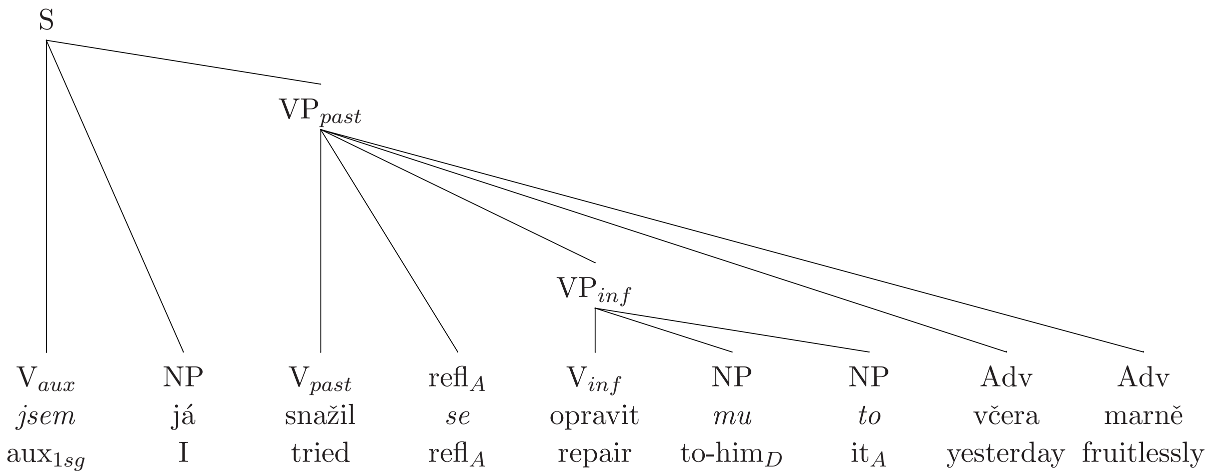
In the formalization, the constraints and functions can refer to the following objects:

- words – the set of words in the sentence;
- a tectogrammatical structure – (i) a set of nonterminals (nodes), (ii) a relation connecting them (dtr), and (iii) a relation connecting a phrase with its head daughter (headDtr);
- a topological tree – (i) a set of topological nodes (tNodes), (ii) a relation connecting these nodes (tDtr), and (iii) a word order relation (wo) between them.

3.4. Topological trees

A topological tree (TT) consists of a set of nodes (tNodes : Set(TNode)) and a relation connecting them into a tree (tDtr). Each node contains an attribute fld – the field the node represents. The constraints imposed on topological trees are of two kinds: (i) those constraining the order of topological fields (e.g., a preclitic field precedes a clitic field); (ii) those constraining the number of occurrences of certain fields – some fields are required, while other fields cannot occur more than once (e.g., every Czech clause has to contain exactly one preclitic field). This is ensured by the following three language independent constraints:

- orderTDtrs constraint. Daughters of every topo node have to satisfy the ordering required for the subfields



Opravit *jsem se mu to* včera snažil marně.

Figure 2: The syntactic structure of (44): ‘I tried to repair it for him yesterday without success.’

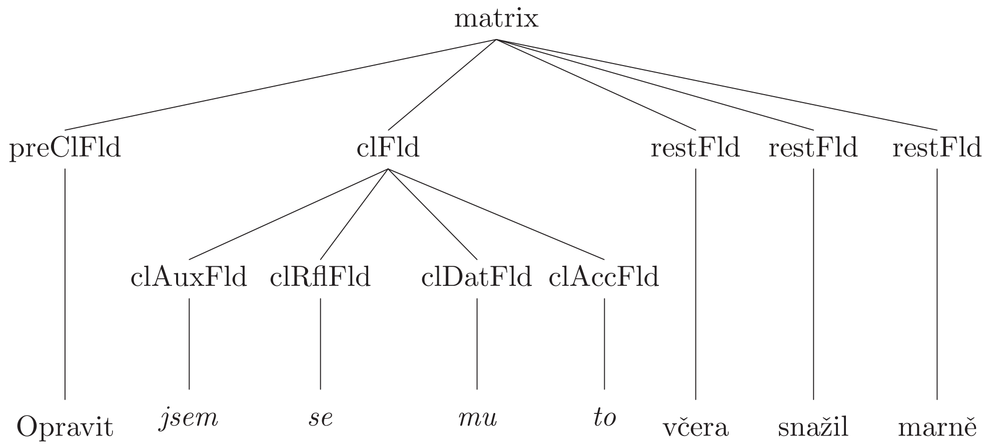


Figure 3: Topological tree of (44): ‘I tried to repair it for him yesterday without success.’

of the field of that node (FLD). This is ensured by the predicate isTopoOrdered.

con TNode.orderTDtrs
  isTopoOrdered(fldsUnder(this), FLD)

- fieldUniqueness constraint.

  Each topo node can have daughters with certain fields only once. Those fields are specified by the language dependent uniqueFields function.

  E.g., There is only one matrix clitic field (clFld).

  con TNode.fieldUniqueness
    $\mathsf{uniqueFields}(\textsc{fld}) \cap \mathsf{duplicates}(\mathsf{fldsUnder}(\mathsf{this})) = \emptyset$

- fieldExistence constraint. Each topo node has at least the daughters with the required fields. The subfields required for a field are specified by the language dependent requiredFields function.

  E.g., Every Czech finite clause contains exactly one preclitic field (preClFld).

  con TNode.fieldExistence
    $\mathsf{requiredFields}(\textsc{fld}) \subseteq \mathsf{toSet}(\mathsf{fldsUnder}(\mathsf{this}))$

The above constraints use (i) several language independent functions specific to topological trees:

- $\mathsf{isTopoOrdered}(\mathit{flds} : \mathsf{List}(\mathsf{Fld}), \mathit{fld} : \mathsf{Fld}) : \mathsf{Bool}$

  The predicate tests whether the list of fields (*flds*) is ordered in the way required for their dominating field (*fld*). The predicate refers to two language dependent functions – prescribedOrder and freeFields – to determine the required order.

  E.g., $\mathsf{isTopoOrdered}(l, \mathsf{clFld})$ is true for $l = \langle \mathsf{clAuxFld}, \mathsf{clAccFld}\rangle$, but not for $l = \langle \mathsf{clAccFld}, \mathsf{clAuxFld}\rangle$.

- fldsUnder(n : TNode) : List(Fld)

  For a given topo node, this returns the list of the fields of its daughters. It uses a HO function map to map the list of daughters to the list of their fields; FLD is an attribute of a topo node, and like every attribute it is a function – in this case, FLD : TNode $\rightarrow$ Fld.

  E.g., if m is the node in Figure 3 that has FLD = clFld, then fldsUnder(m) = ⟨clAuxFld, clRflFld, cDatFld, clAccFld⟩

and (ii) several language-dependent functions:

- prescribedOrder(*fld* : Fld) : glist(Fld)

  Given a field, this returns the list specifying the required order of the subfields within that field. Each member of the list is either a single field or a set of fields; the latter case means that the relative order of the fields within such a set is free.The latter specifies that the relative order of its member is free. See §4.1.1 for a sample instantiation.

- freeFields(*fld* : Fld) : Set(Fld)

  For a given field, this returns the subfields that can occur anywhere within that region. See §4.1.1 for a sample instantiation.

- uniqueFields(*fld* : Fld) : Set(Fld)

  For a given field, this returns the subfields that can occur maximally once within that field (e.g., preclitic or clitic fields in a matrix).

- requiredFields(*fld* : Fld) : Set(Fld)

  For a given field, this returns the subfields that must occur within that field (e.g., a preclitic field in a matrix).

In addition, there are several general purpose functions for working with sets, relations, etc. These functions are often higher order.

4. Clitics in HOG

4.1. Word order within the clitic cluster

I will now formalize some of the constraints described in §2: the constraints on morpholexical ordering in §4.1.1, followed by two constraints on clitic climbing – Continuous Clusters Constraint in §4.1.2 and Ordering by Governors' Degree of Embeddedness Constraint in §4.1.3.

4.1.1. Morpholexical ordering

Since the elements of the clitic cluster are often quite unrelated in terms of phrase structure – they can have different governors that are often far apart – it is not very convenient to express the ordering constraints by referring to phrase structure trees. The topological tree, on the other hand, makes the formalization simple, scalable, and modular (and therefore, for example, relatively easy to port to similar languages).

The support for topological trees developed in §3.4 is flexible enough to capture the above described constraints. The topological tree of every clause with clitics contains a clitic field; such a field further contains sub-fields corresponding to particular positions discussed above (c.f. the topological tree in (3)).

We only have to specify the two language dependent functions prescribedOrder and freeFields to handle the clitic ordering. The prescribedOrder function, given a field, returns the list specifying the required order of its subfields:

prescribedOrder(FLD : Fld) : glist(Fld)

```
  select FLD
    matrix, embedFld : ⟨preClFld, clFld, restFld, finFld⟩
    clFld : ⟨clAuxFld, {clEthDatFld,clRflFld}, clAdjDatFld,
                                     clDatFld, clAccFld, clGenFld⟩
    ⋮
    else  : ⟨⟩
```

The subfields of a particular field have to be in the same order as the fields in the list returned by this function. If a member of the list is a set of fields, the relative order of those fields is free. The freeFields function, given a field, returns the set of subfields that can occur at any position under that field:

```
freeFields(FLD : Fld) : Set(Fld)
  select FLD
    clFld : {clUzFld, clPryFld, clVsakFld, ...}
    ⋮
    else  : {}
```

The functions above specify the order only within the clitic and matrix fields; for a real grammar, it would be necessary to constrain other fields as well, e.g., relative clauses, noun phrases, etc.

4.1.2. Continuous clusters

The Continuous Clusters Constraint stated in (35) and repeated here as (45), can be phrased also as (46).

(45) All clitics in a clitic field with the same governor have to form a continuous cluster.

(46) If clitics in the clitic field are partitioned (grouped) by their governors, then all members of this partitioning are continuous.

In a HO framework, it is also very easy to formalize:

```
con TNode.continuousClusters  $\textsc{fld} = \mathsf{clFld} \Rightarrow$
  $\forall x \in \mathsf{partitionBy}(\mathsf{toOrder}, \mathsf{sameGov})\ .\ \mathsf{isContinuous}(x)$
  where
    $\mathsf{toOrder} = \{w \in \mathsf{wordsUnder}(\mathsf{this}) \mid w.\textsc{lex} \notin \mathsf{freeClitics}\};$
```

The function sameGov is an equivalence predicate – it tests whether two words have the same governor. It is used by the HO function partitionBy to partition the relevant clitic into clusters having the same governor. Each of this clusters has to be continuous. The constraint disregards free clitics like *však* specified by the language dependent function freeClitics. Therefore it takes all the words under the clitic field (wordsUnder(this)) and filters out all the clitics whose lemma is specified as lemma of a free clitic by the function freeClitics, a trivial language dependent function specifying clitics that can occur at any position within the clitic field. The constraint also relies on several general purpose functions for transitive closure (trans, written as $<^+$ or $\leq^*$), finding a minimum (min), etc. . All of them are higher order. See Appendix B for more details.

4.1.3. Ordering by Governors' Degree of Embeddedness Constraint

The Ordering by Governors' Degree of Embeddedness Constraint was stated in plain English in (39) and is repeated here as (47).

(47) Clitics in a clitic field are ordered according to the degree of embedding of their governors: a clitic governed by a less deeply embedded verb precedes a clitic governed by a more deeply embedded verb.

In a HO framework, it is also very easy to express:

```
con TNode.cliticsByGovEmbedding  $\textsc{fld} = \mathsf{clFld} \Rightarrow$
  $\forall x, y \in \mathsf{toOrder}\ .\ \mathsf{mother}(y) <^{+}_{\textsc{dtr}} \mathsf{mother}(x) \Rightarrow x <_{\textsc{wo}} y$
```

where
$\mathsf{toOrder} = \{w \in \mathsf{wordsUnder}(\mathsf{this}) \mid w.\textsc{lex} \notin \mathsf{freeClitics}\}$;

The constraint requires that if a clitic's mother dominates another clitic's mother ($\mathsf{mother}(y) <^{+}_{\mathrm{DTR}} \mathsf{mother}(x)$), the former clitic precedes the later ($x <_{\mathrm{WO}} y$). The constraint disregards free clitics like *však* – see §4.1.2 for comments on the way cliticsToOrder is specified. The constraint relies on the higher order function transitive closure (written as $<^{+}$) to create a transitive closure of the immediate dominance relation (DTR).

5. Summary

This paper presented an analysis of some word order aspects of Czech sentential clitics in a Higher Order Grammar. It showed that Higher Order Grammar, with its powerful type system, higher-order functions and distinction of tectogrammar and phenogrammar, is a convenient tool for expressing constraints over phenomena that are generally nonlocal and involve unbounded dependencies. In the near future, I plan to focus on two issues (i) formalizing the position of the clitic cluster and other word order issues in Czech and (ii) how to make the presented treatment compatible with HOG as presented in (Pollard, 2001b) or (Pollard and Hana, 2003).

Acknowledgement

I thank Carl Pollard, my advisor, for long discussions about HOG and for many suggestions. I also thank Richard Janda, Brian Joseph, Detmar Meurers, Alexander Rosen, Petr Sgall and Andrea Sims for many useful comments.

A. Appendix 1 – HOG Formalism Overview

In this appendix, I provide a brief overview of the HOG formalism. I assume that the reader is familiar with some higher-order logic and I take Ty2 (Gallin, 1975), the higher-order logic most linguists are familiar with, as the point of departure.

A.1. HOG vs. Ty2

Ty2 (Gallin, 1975) is a higher-order logic equivalent to Montague's Intensional Logic. It is an extension of Henkin's logic which in turn is based on Church's Simple Theory of Types. In more detail:

The Simple Theory of Types (Church, 1940) is a logic obtained from typed lambda calculus by moving term equivalence from meta-language to object-language (thus term equivalence can be stated within a theory instead of imposed externally) and by adding constants for logical connectives and quantifiers. There are two basic types: truth values (in HOG called Bool) and entities (Ent), and one type constructor for creating functional types ($\Rightarrow$).[15].

Henkin (1950) provided Church's theory with models (Henkin models). He also added the axiom of propositional extensionality, which says that for truth values, there is no difference between equality and equivalence. He showed that all the logical connectives, constants and quantifiers do not have to be assumed as primitives but are lambda definable.

[15]This constructor is called functional space or exponential; $A \Rightarrow B$ is the type of functions from A into B. In some formalisms, e.g., Montague's IL, $A \Rightarrow B$ is written as $\langle A, B\rangle$

In HOG grammars, I often use the following notation for functional expressions:

simplified notation	full notation
$\mathsf{fnc}(a : \mathsf{Type}) = \mathsf{term}$	$\mathsf{fnc} = \lambda a : \mathsf{Type}\ .\ \mathsf{term}$
$\lambda a, b : \mathsf{Type}\ .\ \mathsf{term}$	$\lambda a : \mathsf{Type}\ \lambda b : \mathsf{Type}\ .\ \mathsf{term}$

Gallin (1975) added one more basic type for possible worlds and showed that the resulting system, Ty2, is equivalent to Montague's IL (Montague, 1974b,a).

The higher order logic of HOG is more powerful than Ty2. HOG has a larger set of basic types, including a type of natural numbers. The type system is polymorphic (although only schematically) and allows definition of separation subtypes. Moreover, while Ty2 has only one type constructor $\Rightarrow$ (defining functional types), HOG has two additional primitive type constructors – products and coproducts. I discuss each of these extensions in more detail below.

A.2. Basic types

The set of basic types in HOG is larger than just the three basic types of Ty2 (Bool,[16] Ent, World). The grammar writer defines the set of basic types, e.g., syntactic categories NP, S, . . . , types of feature values Case, Number, . . .

A.3. Polymorphism

A second-order type system (also known as *parametric polymorphism*) allows abstraction and quantification over types. It is possible to define functions accepting types as parameters and passing them as results. The second-order types, i.e., the "types" of types or sets of types are called *kinds*. This means

1. It is possible to freely extend the set of type constructors. For example, a function (a type operator) can

[16]The type of truth values Bool is a definable type in HOG. Bool = Unit + Unit, where Unit is the nullary cartesian product (§A.4) and + is the cartesian coproduct (§A.5). The two injections ι_0 and ι_1 are then the constants true and false. However, for linguistics, it does not make a difference whether the type is defined or primitive. In Ty2 notation, Bool is called t, Ent e and World s.

return multisets of type A.

2. Functions can be generalized over types. For example, it is possible to define a function min that returns the minimum of any ordered set of the type A.

Many programming languages support various degrees of such polymorphism, e.g., C++ (templates), Java (generics), ML or Haskell.[17] HPSG also has some sort of parametric polymorphism (e.g., *list*(σ) or *set*(σ) in (Pollard and Sag, 1994:396)), although the details have never been spelled out precisely.

Because the requirements for HOG are different from the requirements in programming languages, HOG replaces full polymorphism with schematic polymorphism.[18] That means a grammar written in a polymorphic formalism can be translated into a formalism without polymorphism. This is possible because for every HOG grammar, the number of expressions having a type as a parameter is finite. This approach gives most of the benefits of a polymorphic type-system while avoiding the complexity of models of polymorphically typed logics (hyperdoctrins, see Crole, 1993).

A.4. Products

HOG contains indexed (cartesian) products. The products are similar to records in programming languages like C++ or Pascal. They are a generalization of the usual binary product $A \times B$ that is the type of all pairs $\langle a, b \rangle$, where $a : A$

[17]For a short overview of polymorphism in programming languages, see (Cardelli and Wegner, 1985).

[18]For example, one disadvantage of a programming language using schematic polymorphism (e.g., templates in C++) is that a compiler has to compile the program as a whole – polymorphic modules cannot be fully precompiled. Although this is a serious practical inconvenience for a programming language, for HOG, as a mathematical formalism, this is irrelevant.

and $b : \mathsf{B}$. The indexed products are indexed by a finite set of indices (e.g., natural numbers or a set of suggestive labels like SUBJ, OBJ, etc.). The indexes are grammar specific. The usual binary product is just an indexed product with the indexes being 0 and 1.

There are also term constructors and destructors. The constructors create indexed tuples from terms and the destructors, called *projections*, allow access to the members of such tuples.

(48) **Definition (Products)**
Let J be a finite set of indexes used in the grammar and let $I \subseteq J$, $I = \{i_1, \ldots, i_n\}$. Let $(\mathsf{A}_i)_{i \in I}$ be a family of types, and $(a_i : \mathsf{A}_i)_{i \in I}$ a family of terms, then we can define:

1. An indexed product type:
   $\prod_{i \in I} \mathsf{A}_i$, equivalently $\langle i_1 : \mathsf{A}_1, \ldots, i_n : \mathsf{A}_n \rangle$
2. An indexed tuple:
   $\langle i_1 : a_1, \ldots, i_n : a_n \rangle_{(\mathsf{A}_i)_{i \in I}} : \prod_{i \in I} \mathsf{A}_i$
3. Projections:
   $\pi^j_{(\mathsf{A}_i)_{i \in I}} : \prod_{i \in I} \mathsf{A}_i \Rightarrow \mathsf{A}_j$

The subscript $(\mathsf{A}_i)_{i \in I}$ is usually omitted.

When the set of indexes are natural numbers, we usually write:

1. $\mathsf{A}_0 \times \ldots \times \mathsf{A}_n$ instead of $\langle 0 : \mathsf{A}_0, \ldots, n : \mathsf{A}_n \rangle$.
2. $\langle a_1, \ldots, a_n \rangle$ instead of $\langle 0 : a_1, \ldots, n : a_n \rangle$

When the set of indexes is empty, we get:

1. The nullary product type: $\mathsf{Unit} = \prod_{i \in \emptyset}$
2. The nullary product term: $* : \mathsf{Unit}$
   This serves as a distinguished term.

(49) **Equations (Products)**

(1a) $\pi^j(\langle \ldots, j : e, \ldots \rangle) = e$

(π^j are projections– they pick the right element)

(1b) $\langle i_1 : \pi^{i_1}(p), \ldots, i_n : \pi^{i_n}(p) \rangle = p$

(1c) $\forall a : \mathsf{Unit} \, . \, a = *$

(there is just one term of the type Unit)

A.5. Coproducts (disjoint unions)

Intuitively, a coproduct $A + B$ is a type that can contain terms of type A or of type B. This is similar to partitioning types in HPSG signature. For example, saying $\mathsf{Head} = \mathsf{Substantive} + \mathsf{Functional}$ in HOG is equivalent to saying the type Head is the supertype of the types Substantive and Functional in HPSG.

The formal machinery is slightly more complicated. The formalism requires every term to have exactly one type. A term of a type A cannot also have a type $A+B$. Therefore, the coproduct requires the proper injections: if $a : A$ then $\iota^0_{A+B}(a) : A + B$ and if $b : B$ then $\iota^1_{A+B}(b) : A + B$. Or when viewed from the other side: if $x : A + B$ then either $\exists a : A \, . \, x = \iota^0_{A+B}(a)$ or $\exists b : B \, . \, x = \iota^1_{A+B}(b)$. The injections can be easily omitted because they are retrievable from the context.[19]

The notion of coproducts, or disjoint unions, is dual to the notion of products. In programming languages, they

[19]They are not uniquely retrievable for "nonlinear" coproducts of the form $A + A$ (more than once the same type). However, I cannot think of any linguistic motivation for such types.

The only type of such form is $\mathsf{Bool} = \mathsf{Unit} + \mathsf{Unit}$. But that's more a consequence of the formal game of trying to assume as few primitive types as possible. For linguistics, it would not make any difference if Bool would be a primitive type and the "nonlinear" products were prohibited.

have various, usually less universal or abstract, counterparts. For example, switch/case statements in Java/C++ are very similar. But so are, from other point of view, unions in C++ or variant records in Pascal. In the case of products, one can see tuples as datastructures and projections as programs (although trivial) for accessing the data. In the case of coproducts, it is the other way round – the injections represent (tagged) data structures and co-tuples programs for accessing and manipulating the data.

(50) **Definition (Coproducts)**
Let J be a finite set of indexes used in the grammar and let $I \subseteq J$, $I = \{i_1, \ldots, i_n\}$. Let $(\mathsf{A}_i)_{i \in I}$ be a family of types, and $(a_i : \mathsf{A}_i)_{i \in I}$ a family of terms, then we can define:

- An indexed coproduct type: $\coprod_{i \in I} \mathsf{A}_i$, equivalently $(i_1 : \mathsf{A}_1, \ldots, i_n : \mathsf{A}_n)$
- An indexed co-tuple: $(i_1 : a_1, \ldots, i_n : a_n)_{(\mathsf{A}_i)_{i \in I}} : \coprod_{i \in I} \mathsf{A}_i$
- Injections: $\iota^j_{(\mathsf{A}_i)_{i \in I}} : \mathsf{A}_j \Rightarrow \coprod_{i \in I} \mathsf{A}_i$

The subscript $(\mathsf{A}_i)_{i \in I}$ is usually omitted.

When the set of indexes are natural numbers, we usually write:

- $\mathsf{A}_0 + \ldots + A_n$ instead of $(0 : \mathsf{A}_0, \ldots, n : \mathsf{A}_n)$.
- $(a_1, \ldots, a_n)$ instead of $(0 : a_1, \ldots, n : a_n)$

When the set of indexes is empty, we get:

- The nullary coproduct type: $\mathsf{Zero} = \coprod_{i \in \emptyset}$
- There is no term of the type Zero

(51) **Equations (Coproducts)**

(2a) $(\langle \ldots, j : e, \ldots \rangle)\iota^j = e$

ι^j are injections.

A.6. Predicate subtyping

HOG supports a very powerful kind of subtyping. For every type A and every predicate $\phi : A \rightarrow \mathsf{Bool}$,

(52) $\lceil A \mid \phi(\mathsf{this}) \rceil$

is a type.[20] Therefore for every type it is possible to define a subtype, whose members have a certain property. For example, if NP is a type we can define the type of accusative NP's as $\mathsf{NPacc} = \lceil \mathsf{NP} \mid \textsc{case}(\mathsf{this}) = \mathsf{acc} \rceil$

The formalism has a monotyping property – every object is a member of exactly one type. That means that functions defined for a particular type cannot be applied to objects of subtypes of that type (e.g., a function defined for $\mathbb{N}$ cannot be applied to the type of even natural numbers). However, there is an easy way around this limitation – for every type A, B where B is a subtype of A, there is an inclusion function $\mu_{B,A} : B \rightarrow A$. Let $g : A \rightarrow C$, then g can be

- directly applied to objects of type A: $g(x)$ for $x : A$.
- indirectly applied to objects of type B via $\mu_{B,A}$: $g(\mu_{B,A}(y))$ for $y : B$.

Sugar: Since for a particular grammar the inclusion function μ can be derived from context, it is usually omitted – we write $g(y)$ instead of $g(\mu_{B,A}(y))$ for $g : A \rightarrow B$ and $y : B$. The formalism still has the monotyping property.

In addition to the notation in (52), a subtype can be defined as:

(53) $\mathsf{type}\ B : A$
    $\mathsf{con}\ \phi_1(\mathsf{this})$

[20] A common notation for predicate types is $\lceil x : A \mid \phi(x) \rceil$. Here, the designated variable name (this) is used to make it notationally simple to split the predicate into several subpredicates, c.f. (53)

$\vdots$

$\mathsf{con}\ \phi_n(\mathsf{this})$

where each ϕ_i is called a constraint and $\phi = \phi_1 \wedge \ldots \phi_n$. To increase readability, a constraint can be given a name; in such case the form is $\mathsf{con\ name}\ \phi_i(\mathsf{this})$.

A.7. Natural Numbers and induction

The logic contains a type of natural numbers, written as $\mathbb{N}$, as a primitive type. Unlike the other primitive types, this is an infinite type. Thus, adding the type is equivalent to adding the axiom of infinity.

The type goes with several terms: numbers (zero and the successor function) and a primitive recursor for induction:

(54) **Definition (Terms on $\mathbb{N}$)**

$\mathsf{0} : \mathbb{N}$

$\mathsf{succ} : \mathbb{N} \Rightarrow \mathbb{N}$

$\mathsf{ind} : \mathsf{A} \times (\mathbb{N} \times \mathsf{A} \Rightarrow \mathsf{A}) \times \mathbb{N} \Rightarrow \mathsf{A}$

We write 1 for $\mathsf{succ}(0)$, 2 for $\mathsf{succ}(\mathsf{succ}(0))$, etc.

(55) **Equations (Induction)**

$\mathsf{ind}(x, f, 0) = x$

$\mathsf{ind}(x, f, \mathsf{succ}(n)) = f(n, \mathsf{ind}(x, f, n))$

There are also the usual Peano axioms.

(56) **Example (Induction)**

Using the induction function, it is possible to define many recursive functions. As an example I show how to define addition and factorial.

- Addition. Addition can be recursively defined in the following way:

  (3a) $\mathsf{add}(i, 0) = i$

  (3b) $\mathsf{add}(i, j) = \mathsf{succ}(\mathsf{add}(i, j - 1))$

  Thus, in HOG we can define the function as:

  (3c) $\mathsf{add} = \lambda i, j : \mathbb{N} \,.\, \mathsf{ind}(i, \lambda n, k : \mathbb{N} \,.\, \mathsf{succ}(k), n)$

  The variable k of the induction steps corresponds to $\mathsf{add}(i, j - 1)$. Note that the induction step ignores the depth of recursion (n): 1 is always added regardless whether it is the first or 25th addition.

  Multiplication is analogous.

- Factorial.

  (3d) $\mathsf{factorial}(0) = 1$

  (3e) $\mathsf{factorial}(j) = j * \mathsf{factorial}(j - 1)$

  Thus the function can be defined as:

  (3f) $\mathsf{factorial} = \lambda j : \mathbb{N} \,.\, \mathsf{ind}(1, \lambda n, k : \mathbb{N} \,.\, n * k, j)$

  The variable k of the induction steps corresponds to $\mathsf{factorial}(j-1)$. This time we cannot ignore the depth of recursion – when called for the 1st time it multiplies by 1, when for the 25th time, by 25:

A.8. Sets

Sets are simulated via their characteristic function.

(57) a. Types
$\mathsf{set} = a \rightarrow \mathsf{Bool}$ (arity = 1)

b. Terms

$\in : a, \mathsf{set}(a) \rightarrow \mathsf{Bool}$ (infix)

Sing[21] $: \mathsf{set}(a) \rightarrow a$ (prefix)

c. Sugar

$\{e_1, \dots e_n\}_A$	for $\lambda x : A \,.\, x = e_1 \vee \dots \vee x = e_n$
$\emptyset_A = \{\}_A$	for $\lambda x : A \,.\, x \neq x$
$\{x : A \in E \mid \phi(x)\}$	for $\lambda x : A \,.\, (E(x) \wedge \phi(x))$
$\forall x : A \in E.\ \phi(x)$	for $\forall x : A \,.\, x \in E \Rightarrow \phi(x)$
$\exists x : A \in E.\ \phi(x)$	for $\exists x : A \,.\, x \in E \wedge \phi(x)$

The subscript/typing A is usually omitted if clear from context.

d. Equations

$e \in s = s(e)$

$\mathsf{Sing}\ \{x\} = x$

A.9. Other features

Type definitions

There is a syntactic sugar for defining types and functions defined on those types:

$\mathsf{type}\ A$
  $\mathsf{attr}_1 : F_1$
  $\vdots$
  $\mathsf{attr}_n : F_n$

defines a type A and functions $\mathsf{attr}_i = F_i$; the type of attr_i is $A \rightarrow X_i$ where X_i is the type of F_i.

if – then – else

Using boolean connectives and singularizer, we can define if – then – else construct:

[21] This is Russell's iota operator, used for example in the analysis of definite articles.

$\mathsf{if}\ \phi\ \mathsf{then}\ E\ \mathsf{else}\ F$ means $\mathsf{Sing}\ \{x \in \{E, F\} \mid (\phi \Rightarrow x = E) \wedge (\neg\phi \Rightarrow x = F)\}$

that means if ϕ is true, the whole construct has the value of E, otherwise it has the value of F. For multiple matching tests, it's better to use select:

select E	means	if $E = E_1$ then F_1
$E_1 : F_1$		$\vdots$
$\vdots$		else if $E = E_n$ then F_n
$E_n : F_n$		else F
else : F		

where

Every function or constraint can contain local definitions of functions or assignments to variables. These local definitions are placed at the end of the function/constraint each on a separate line and are introduced by the keyword where.

Relations

Similarly as sets, relations are modeled via their characteristic functions.

(58) Sugar
$E <_\rho F$ means $\rho(E, F) \wedge E \neq F$
$E \leq_\rho F$ means $\rho(E, F) \wedge E = F$

A.10. Lists (Kleene star)

Lists are defined as functions from natural numbers (indexes) to the type of the elements. However, since HOG allows only total functions and does not have dependent types, such a function could be directly used only for infinite lists. The finite lists are then defined as equivalence classes on those infinite lists where members are considered only up to a certain point.[22]

This means that lists are lambda-definable within HOG. Of course, users of the formalism do not have to care about the way in which the type constructor List is defined and they can use it as if it were a primitive type. Any computational implementation of HOG would also implement lists directly as primitives.

(59) **List types**

List	(type constructor, arity = 1)
$*$	(equivalent notation, Kleene-star)

[22]The polymorphic type $\mathsf{List}(A)$ is defined in two steps. First, the auxiliary (polymorphic) type $\mathsf{Prelist}$:

(4a) $\mathsf{Prelist}(A) = \langle \textsc{iList} : \mathbb{N} \Rightarrow A, \textsc{len} : \mathbb{N} \rangle$

Members of this type are pairs where iList is an infinite list and len is the length of the modeled list. The real lists are defined as equivalence classes on $\mathsf{Prelists}$, where the irrelevant elements (i.e., anything beyond len) are ignored. Below is the corresponding equivalence relation – it considers prelists equivalent if they have the same relevant elements:

(4b)
$$\begin{aligned}&\mathsf{same}(l_1 : \mathsf{Prelist}(A), l_2 : \mathsf{Prelist}(A)) : \mathsf{Bool} = \\ &\quad \textsc{len}(l_1) = \textsc{len}(l_2) \\ &\quad \forall i \in \textsc{len}(l_2)\ .\ \textsc{iList}(l_1)(i) = \textsc{iList}(l_2)(i)\end{aligned}$$

Now it is possible to define the real type constructor:

(4c)
$$\begin{aligned}&\mathsf{List}(A) = [x : \mathsf{Set}(\mathsf{Prelist}(A)) \mid \\ &\quad \exists q \in x\ \forall p : \mathsf{Prelist}(A)\ .\ p \in x \Leftrightarrow \mathsf{same}(q, p)]\end{aligned}$$

and basic functions for working with lists:

(4d) $\mathsf{length}(l : \mathsf{List}(A)) : \mathbb{N} = \mathsf{Sing}\ \{\textsc{len}(k) \mid k \in l\}$

(4e) $\mathsf{itemAt}(l : \mathsf{List}(A), i : \mathbb{N}) : A = \mathsf{Sing}\ \{\textsc{iList}(k)(i) \mid k \in l\}$

and finally some syntactic sugar for specifying lists:

(4f) $[e_1, \ldots e_n]_A$ means $\{u : \mathsf{Prelist}(A) \mid \textsc{len}(u) = n \wedge \textsc{iList}(u)(i) = e_i\}$

(60) **List terms**

$[e_1, \ldots e_n]_A$	: $\mathsf{List}(A)$	(list of length n)
len	: $\mathsf{List}(A) \Rightarrow \mathbb{N}$	(list length)
itemAt	: $\mathsf{List}(A) \times \mathbb{N} \Rightarrow A$	(indexed list access)
los	: $A \Rightarrow \mathsf{List}(A)$	(singleton list)

A.11. Semantics

Since HOG is a standard HO logic, we get the semantics of the theory for free. The model is a topos (i.e., categorical generalization of sets) – see (Lambek and Scott, 1986) or (Crole, 1993) for more details.

B. Appendix 2 – Clitics in HOG Overview

With the exception of compaction (preClFld, NP/PP), this fragment captures the same constraints as Penn (1999), but in a much more modular and compact way. For example we do not have to write 10 different uniqueFields constraints, it is easily modifiable for another language, and most of the functions are used in other areas of grammar, too (maxProjection, min, etc). In addition, the fragment contains the Continuous Clusters Constraint (35) and the Ordering by Governors' Degree of Embeddedness Constraint (39).

Language specific

- continuousClusters constraint. If clitics in a clitic field are partitioned by their governors, then all members of this partitioning are continuous.

  con TNode.continuousClusters $\text{FLD} = \mathsf{clFld} \Rightarrow$
  $\forall x \in \mathsf{partitionBy}(\mathsf{toOrder}, \mathsf{sameGov})\ .\ \mathsf{isContinuous}(x)$
  where

```
    toOrder = $\{w \in$ wordsUnder(this) $|$
                                        $w.\textsc{lex} \notin$ freeClitics$\}$;
```

- cliticsByGovEmbedding constraint. In a clitic field, a clitic governed by a less deeply embedded verb precedes a clitic governed by a more deeply embedded verb. This is a stronger version of the cliticsByGovs constraint.

```
con TNode.cliticsByGovEmbedding   $\textsc{fld}$ = clFld $\Rightarrow$
  $\forall x, y \in$ toOrder .
                mother($y$) $<^{+}_{\mathrm{DTR}}$ mother($x$) $\Rightarrow x <_{\mathrm{WO}} y$
  where
    toOrder = $\{w \in$ wordsUnder(this) $|$
                                        $w.\textsc{lex} \notin$ freeClitics$\}$;
```

- freeClitics. A trivial function specifying those clitic that can occur anywhere in a clitic field.

freeClitics = {však, . . . }

- prescribedOrder function. Given a field, it returns the list specifying the required order of its subfields. The subfields of a particular field have to be in the same order as the fields in the list returned by this function. If a member of the list is a set of fields, the relative order of those fields is free.

```
prescribedOrder($\textsc{fld}$ : Fld) : glist(Fld)
  select $\textsc{fld}$
    matrix, embedFld : $\langle$preClFld, clFld, restFld, finFld$\rangle$
    clFld : $\langle$clAuxFld, $\{$clEthDatFld,clRflFld$\}$,
              clAdjDatFld,clDatFld, clAccFld, clGenFld$\rangle$
    ⋮
    else  : $\langle\rangle$
```

- freeFields function. Given a field, it returns the set of subfields that can occur at any position under that field.

```
freeFields(FLD : Fld) : Set(Fld)
  select FLD
    clFld : {clUzFld, clPryFld, clVsakFld, ...}
    ⋮
    else  : {}
```

- uniqueFields function. Fields that can occur only once in a particular bush.

  uniqueFields() : Set(Fld) = {preClFld, clFld, cRflFld, ...}

- requiredFields. Fields that are required within a particular bush.

  ```
  requiredFields(f : Fld) : Set(Fld)
    select FLD
      matrix, embedFld : {preClFld}
      ⋮
      else  : {}
  ```

Topological trees specific

- orderTDtrs constraint. Daughters of every topo node have to satisfy the ordering imposed by the predicate isTopoOrdered

  ```
  con TNode.orderTDtrs
    isTopoOrdered(fldsUnder(this), FLD)
  ```

- matrixCompaction constraint. Matrix must be compacted.

  ```
  con matrixCompaction
    $\forall x \in \text{TNODES} \,.\, x \leq^{*}_{\text{TDTR}} \text{this} \Leftrightarrow \text{this.FLD} = \text{matrix}$
  ```

- fieldUniqueness constraint. Each topo node can have daughters with certain fields only once. Those fields are specified by the language dependent uniqueFields function.

  ```
  con TNode.fieldUniqueness
    uniqueFields(FLD) ∩ duplicates(fldsUnder(this)) = ∅
  ```

- fieldExistence constraint. Each topo node has at least the daughters with required fields. Those required subfields are specified by the language dependent requiredFields function.

  con TNode.fieldExistence
  $\quad$ requiredFields($\textsc{fld}$) $\subseteq$ toSet(fldsUnder(this))

- fldsUnder function. Function returning the list of all the fields under a node.

  fldsUnder(n : TNode) : List(Fld) = map($n.\textsc{dtrs}, \textsc{fld}$)

- region function. The lowest node dominating all specified words.

  region(*words* : List(Word)) : TNode =
  $\quad \min_{\text{TDTR}}\{h \mid \forall w \in \mathit{words}.w <^{+}_{\text{TDTR}} h\}$

- isTopoOrdered function. This predicate tests whether a list of fields (*flds*) is ordered in the way required for their dominating field (*fld*). The predicate refers to two language dependent functions – prescribedOrder and freeFields to determine the required order.

  isTopoOrdered(*flds* : List(Fld), *fld* : Fld) =
  $\quad$ imposeOrder($flds$, prescribedOrder($\textsc{fld}$),
  $\qquad\qquad$ freeFields($\textsc{fld}$))

- imposeOrder. This tests whether the list l is properly ordered. The argument *ord* specifies a partial order on elements; the argument *free* is a set of elements that can occur anywhere. If two items in the l list are ordered in a certain way, then they either have to be ordered the same way in the *ord* list or at least one of them has to be in the *free* set. This is a polymorphic function ordering lists with members of any type, but we use it to order a list of fields.

  imposeOrder(l: List(a), *ord*: glist(a), *free*: Set(a)): Bool
  $\quad \forall i, j \in$ indexes(l) . $i < j \Rightarrow$

```
                    $(l[i] <_r l[j] \vee l[i] \in \mathit{free} \vee l[j] \in \mathit{free})$
  where $r = \mathsf{partialOrder}(\mathit{ord})$
```

- topoField. The field of the word w relative to the topo node h.

```
topoField($w$ : Word, $h$ : TNode) : TNode
  if $(w <_{\mathrm{TDTR}} h)$
    then $h$
    else Sing $\{hDtr | w <^{+}_{\mathrm{TDTR}} hDtr <_{\mathrm{TDTR}} h\}$
```

- wordsUnder. The set of words under the daughter of a node with the specified field.

```
wordsUnder($n$ : TNode, $f$ : Fld) : Set($a$)
  wordsUnder(Sing $\{x \mid x <_{\mathrm{TDTR}} n \wedge x.\mathrm{FLD} = f\}$)
```

- wordsUnder. The set of words under a node.

```
wordsUnder($n$ : TNode) : Set($a$)
  $\{w$ : Word $\mid w \leq^{*}_{\mathrm{TDTR}} n\})$
```

Tecto-structure specific

- sameGov $= \lambda a, b$: Word . gov$(a) =$ gov(b)

- gov. Governor of a word.

```
gov : Word $\rightarrow$ Word =
  maxProjection $\circ$ mother $\circ$ lexHead
```

- maxProjection. Maximal projection of a word.

```
maxProjection($w$ : Word) : Node =
  $\max_{\mathrm{HEADDTR}}\{h \mid w \leq^{*}_{\mathrm{HEADDTR}} h\}$
```

- lexHead. Lexical head of a phrase.

```
lexHead($p$ : Node) : Word =
  Sing $\{w$ : Word $\mid w <^{+}_{\mathrm{HEADDTR}} p\}$
```

General Purpose functions

- transClosure. Transitive closure of a relation; it is written as $<^{+}$ or $\leq^{*}$.

  $\mathsf{transClosure}(r : \mathsf{Rel}(a)) : \mathsf{Rel}(a) =$ (HO)
  $\lambda x, y : a \;.\; \exists n : \mathbb{N} \;.\; r^{\mathsf{n}}(x, y)$

- min. Minimum of a set relative to a relation.

  $\min(r : \mathsf{Rel}(a), s : \mathsf{Set}(a)) : a =$ (HO)
  $\mathsf{Sing}\ \{x \in s \mid \forall y \in s \;.\; y \neq x \Rightarrow x <_r y\}$

- map. Transforms a list to another list using a function. E.g., $\mathsf{map}([1, 2, 3], \lambda x \;.\; x + 1) = [2, 3, 4]$

  $\mathsf{map}(l : \mathsf{List}(a), f : a \rightarrow b) : \mathsf{List}(b) =$ (HO)
  $\mathsf{Sing}\ \{k : \mathsf{List}(b) \mid \forall i \in \mathsf{indexes}(l) :$
  $k[i] = f(l[i]) \wedge \mathsf{length}(l) = \mathsf{length}(k)\}$

- duplicates. Returns the set of all the members of a list that occur in the list more than once.
  E.g., duplicates([1,1,2,3,3]) = {1,3}

  $\mathsf{duplicates}(l : \mathsf{List}(a)) =$
  $\{x \mid \exists i, j \in \mathsf{indexes}(l) \;.\; i \neq j \wedge x = l[i] = l[j]\}$

- indexes. The set of indexes of a list.

  E.g., indexes([1,1,2,3]) = {0,1,2,3}

  $\mathsf{indexes}(l : \mathsf{List}(a)) : \mathsf{Set}(\mathbb{N}) =$
  $\{i : \mathbb{N} \mid 0 \leq i < \mathsf{length}(l)\}$

- partitionBy. A HO function partitioning a set into equivalence classes by the relation *eq*.

  $\mathsf{partitionBy}(s : \mathsf{Set}(a), eq : \mathsf{Rel}(a)) : \mathsf{Set}(\mathsf{Set}(a)) =$
  $\{\mathsf{eqClass}(x, s, eq) \mid x \in s\}$ (HO)

- eqClass. A HO function giving the equivalence class containing the element x.

  $\mathsf{eqClass}(x : a, s : \mathsf{Set}(a), eq : \mathsf{Rel}(a)) : \mathsf{Set}(a) =$ (HO)
  $\{y \in s \mid eq(x, y)\}$

- filter. A HO function filtering a set using the predicate ϕ.

  $\mathsf{filter}(s : \mathsf{Set}(a), \phi : a \to \mathsf{Bool}) : \mathsf{Set}(a) =$ (HO)
  $\{x \in s \mid \phi(x)\}$

- toSet. Creates a set from a list.

  E.g., $\mathsf{toSet}([1,1,2,3]) = \{1,2,3\}$

  $\mathsf{toSet}(l : \mathsf{List}(a)) : \mathsf{Set}(a) = \{x \in l\}$

- $\mathsf{partialOrder}$. Creates a partial order based on a glist

  $\mathsf{partialOrder}(l : \mathsf{glist}(a)) : \mathsf{Rel}(a) =$ (HO)
  $\mathsf{Sing}\ \{\rho \mid \forall x, y\ .\ \rho(x,y) \Leftrightarrow$
  $\exists x', y'\ .\ \mathsf{isSublistOf}([x', y'], l) \wedge$
  $(x = x' \vee x \in x') \wedge (y = y' \vee y \in y')\}$

- $\mathsf{isSublistOf}$. Checks whether one list is a sublist of another list. The elements of the sublist must be adjacent. E.g., $\mathsf{isSublistOf}([2,3],[1,2,3,4]) = \mathsf{true}$

  $\mathsf{isSublistOf}(la : \mathsf{List}(a), lb : \mathsf{List}(a)) : \mathsf{Bool}$
  $\exists i \in \mathsf{indexes}(lb) \forall j \in \mathsf{indexes}(la) : la[j] = lb[i+j]$

- glist type. List containing elements or set of elements

  $\mathsf{type}\ \mathsf{glist}(a) = \mathsf{List}(a + \mathsf{Set}(a))$

- Rel(a). Polymorphic relation type.

  $\mathsf{type}\ \mathsf{Rel}(a) = a \times a \to \mathsf{Bool}$

References

Bémova, Alena, Jan Hajič, Barbora Hladká and Jarmila Panevová (1999) "Morphological and Syntactic Tagging of the Prague Dependency Treebank". *Proceedings of ATALA Workshop*, Paris, France, 21–29.

Cardelli, Luca and Peter Wegner (1985) "On understanding types, data abstraction, and polymorphism". *Computing Surveys* 17(4), 471–522.

Church, Alonzo (1940) “A Formulation of the Simple Theory of Types”. *The Journal of Symbolic Logic* 5(2), 56–68.

Crole, Roy L. (1993) *Categories for Types*. Cambridge University Press.

Curry, Haskell B. (1961) “Some Logical Aspects of Grammatical Structure”. Roman Jakobson, ed., *Structure of Language and Its Mathematical Aspects*, 56–68.

Dowty, David R. (1996) “Toward a minimalist theory of syntactic structure”. Harry Bunt and Arthur van Horck, eds., *Discontinuous Constituency*, Berlin, New York: Mouton de Gruyter, 11–62.

Drach, Erich (1937) *Grundgedanken der deutschen Satzlehre*. Frankfurt: Diesterweg. 4th edition, Darmstadt: Wissenschaftliche Buchgesellschaft, 1963.

Franks, Steven and Tracy Holloway King (2000) *A Handbook of Slavic Clitics*. Oxford University Press.

Gallin, D. (1975) *Intensional and Higher Order Modal Logic*. Amsterdam: North Holland.

Henkin, L. (1950) “Completeness in the theory of types”. *Journal of Symbolic Logic* 15, 81–91.

Holan, Tomáš, Vladislav Kuboň, Karel Oliva and Martin Plátek (1998) “Two useful measures of word order complexity”. Sylvain Kahane and Alain Polguère, eds., *Proceedings of COLING-ACL ’98 Workshop “Processing of Dependency-Based Grammars”*, Montreal: University of Montreal.

Karlík, Petr, Marek Nekula and Z. Rusínová (1996) *Příruční mluvnice češtiny*. Praha: Nakladatelství Lidové Noviny.

Kathol, Andreas (1995) “Linearization-Based German Syntax”. Ph.D. thesis, The Ohio State University.

Kepser, Stephan (2001) “On the Complexity of RSRL”. Geert-Jan Kruijff, Lawrence S. Moss and Richard T. Oehrle, eds., *Proceedings FG-MOL 2001*. Elsevier Science, no. 53 in Electronic Notes in Theoretical Computer

Science, 43–52.

Lambek, Joachim (1988) “Categorial and categorical grammars”. R. Oehrle, E. Bach and D. Wheeler, eds., *Categorial Grammars and Natural Language Structures*, Dordercht: Reidel, 297–317.

______ (1999) “Deductive systems and categories in linguistics”. H. Ohlbach and U. Reyle, eds., *Logic, Language, and Reasoning*, Dordercht: Kluwer, 279–294. Essays in Honor of Dov Gabbay.

Lambek, Joachim and P. J. Scott (1986) *Introduction to Higher Order Categorical Logic*. Cambridge University Press.

Milner, Robin, Mads Tofte, Robert Harper and David MacQueen (1997) *The Definition of Standard ML (Revised)*. Cambridge, MA: MIT Press.

Montague, R. (1974a) “English as a formal language”. 188–221.

______ (1974b) “The proper treatment of quantification in ordinary English”. R. Thomason, ed., *Formal Philosophy*, New Haven, CT: Yale University Press, 247–270.

Oliva, Karel and Tania Avgustinova (1995) *The position of Sentential Clitics in the Czech Clause*. CLAUS Report 68, Universität des Saarlandes.

Penn, Gerald (1999) “A Generalized-Domain-Based Approach to Serbo-Croatian Second-Position Clitic Placement”. Gosse Bouma, Erhard Hinrichs, Geert-Jan M. Kruijff and Richard Oehrle, eds., *Constraints and Resources in Natural Language Syntax and Semantics*, Stanford: CSLI Publications, 119–136.

Pollard, Carl (2001a) “Cleaning the HPSG Garage: Some Problems and some proposals”.

______ (2001b) “Higher-order grammar: a categorical foundation for type-logical constraint-based grammar”. *Proceedings of Formal Grammar*. Helsinki.

Pollard, Carl and Jiri Hana (2003) “Ambiguity, neutrality, and coordination in higher order grammar". Gerhard Jaeger, Paola Monachesi, Gerald Penn and Shuly Wintner, eds., *Proceedings of Formal Grammar*. Wien, 125–136.

Pollard, Carl J. and Ivan A. Sag (1994) *Head-Driven Phrase Structure Grammar*. Chicago: University of Chicago Press.

Reape, Mike (1994) "Domain Union and Word Order Variation in German". John Nerbonne, Klaus Netter and Carl J. Pollard, eds., *German in Head-Driven Phrase Structure Grammar*, Stanford University: CSLI Publications, no. 46 in CSLI Lecture Notes, 151–197.

Richter, Frank (2000) "A Mathematical Formalism for Linguistic Theories with an Application in Head-Driven Phrase Structure Grammar". Dissertation, Eberhard-Karls-Universität Tübingen, Version of April 28th, 2000. Version of April 28th, 2000. Superseded by Richter 2004.

Rosen, Alexandr (2001) "A constraint-based approach to dependency syntax applied to some issues of Czech word order". Ph.D. thesis, Charles University, Prague.

Sgall, Petr, Eva Hajičová and Jarmila Panevová (1986) *The Meaning of the Sentence and Its Semantic and Pragmatic Aspects*. Prague, Czech Republic/Dordrecht, Netherlands: Academia/Reidel Publishing Company.

Sgall, Petr, Ladislav Nebeský, Alla Goralčíková and Eva Hajičová (1969) *A Functional Approach to Syntax in Generative Description of Language*. New York: American Elsevier Publishing Company.

Toman, Jidřich (1996) "A note on clitics and prosody". Aaron L. Halpern and Arnold M. Zwicky, eds., *Approaching second. Second position clitics and related phenomena*, Stanford, California: CSLI Publications, no. 61 in CSLI Lecture Notes, 505–510.

Wackernagel, Jacob (1892) "Über ein Gesetz der indogermanischen Wortstellung". *Indogermanische Forschungen*

1, 333–436.