

Gramatické formalismy pro ZPJ II

Aleš Horák

E-mail: hales@fi.muni.cz

http://nlp.fi.muni.cz/poc_lingv/

Obsah:

- ▶ HPSG – Head-driven Phrase Structure Grammar
- ▶ SET – pravděpodobnostní závislostní gramatika
- ▶ Metagramatika systému synt

HPSG – Head-driven Phrase Structure Grammar

- ▶ HPSG, **Head-driven Phrase Structure Grammar** – Pollard & Sag, 1994
- ▶ navazuje na Gazdar, **Generalized Phrase Structure Grammar**, 1985
- ▶ **lexikalizovaná** teorie generativní gramatiky přirozeného jazyka
- ▶ *neterminály* CFG jsou nahrazeny **příznakovými strukturami**
- ▶ založená na **omezeních** (constraints)
- ▶ modeluje jazyk pomocí **deklarativních omezení** typovaných struktur. Pro vyhodnocení omezení se používá **unifikace** mezi příznakovými strukturami.
- ▶ **příznaky** jsou propojeny pomocí **strukturního sdílení**, tedy předáváním proměnných mezi podstrukturami dané struktury
- ▶ HPSG je **nederivační**, na rozdíl od jiných formalismů, kde jsou různé úrovně syntaktické struktury sekvenčně odvozovány pomocí transformačních operací

HPSG – Head-driven Phrase Structure Grammar – pokrač.

- ▶ gramatika je v HPSG modelována pomocí **uspořádaných příznakových struktur**, které korespondují s typy výrazů přirozeného jazyka a jejich částmi
- ▶ cílem teorie je detailní specifikace, které příznakové struktury jsou **přípustné**
- ▶ příznakové struktury definují **omezení**
hodnoty příznaků mohou být jednoho ze čtyř typů
 - atomy
 - příznakové struktury
 - množiny příznakových struktur ($\{\dots\}$)
 - nebo seznamy příznakových struktur ($\langle\dots\rangle$)

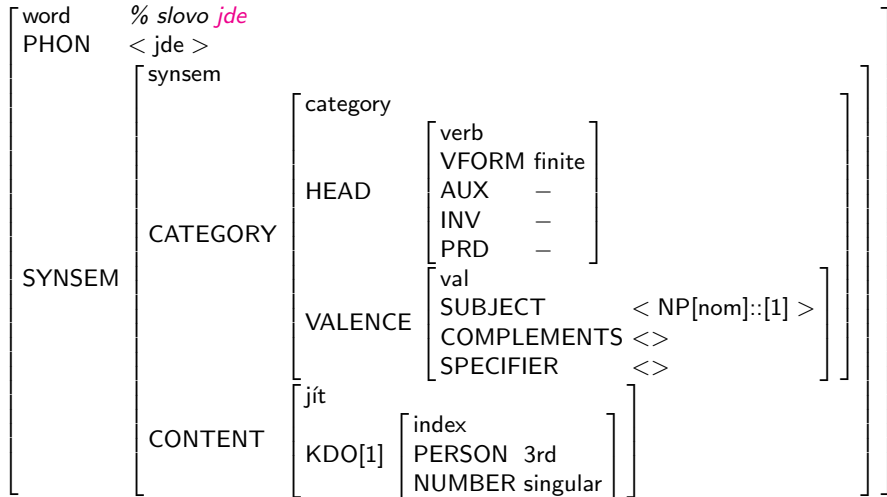
HPSG – lexikální hlava

- ▶ **slova** (lexikální položky) obsahují **hodně informací** – podle psycholingvistiky se podobá *zpracování v lidském mozku*
- ▶ **lexikální hlava** – základní prvek frázové struktury HPSG
lexikální hlava = jedno slovo, jehož položka specifikuje informace, které určují základní gramatické **vlastnosti fráze**, kterou hlava zastupuje
gramatické vlastnosti zahrnují:
 - morfologické informace (part-of-speech, POS)
N zastupuje NP, VP zastupuje S, V zastupuje VP
 - relace závislosti (např. valenční rámec slovesa)
- ▶ lexikální hlava obsahuje také klíčové **sémantické informace**, které sdílí se zastupovanou frází

HPSG – struktury

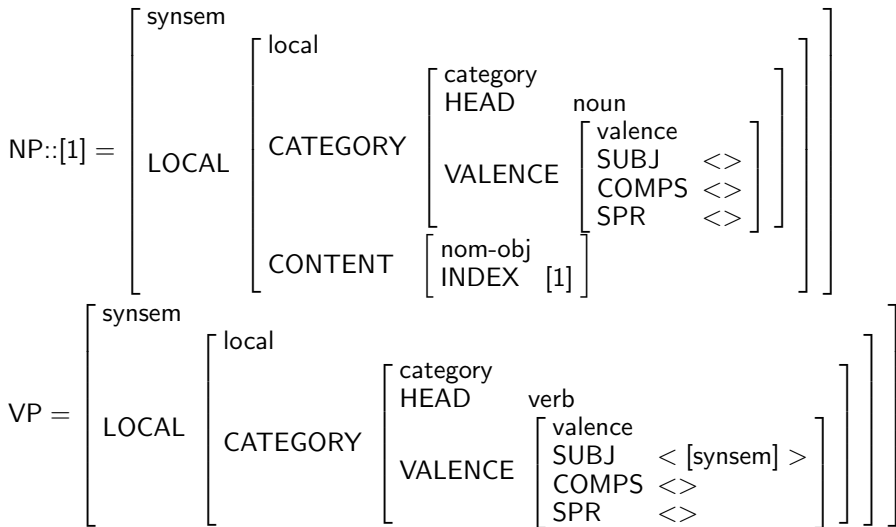
HPSG struktury jsou **typované příznakové struktury**

zapisují se pomocí AVM – **příznaky** velkými písmeny, **typy** malými



HPSG – syntaktické kategorie

symbolsy **syntaktických kategorií** – zkratky určitých příznakových popisů:



HPSG – lexikální položky

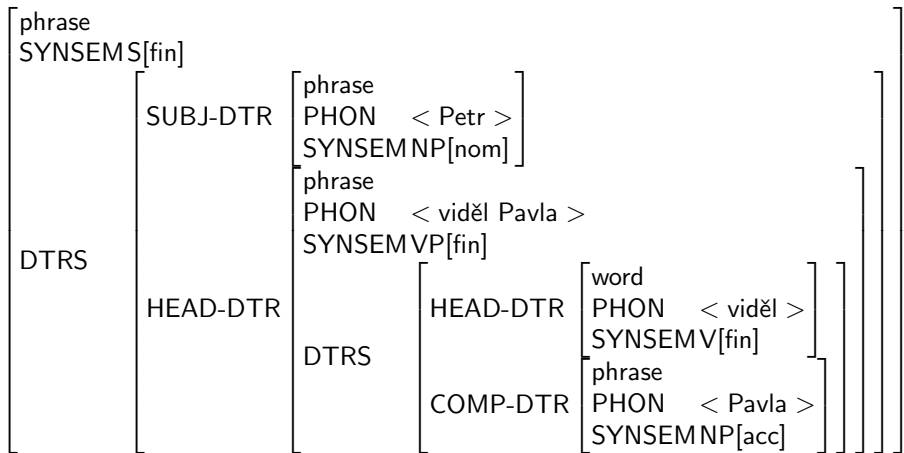
velké množství akcí je v **lexikonu**:

JÍT	CATEGORY	HEAD	verb					
		VALENCE	<table> <tr> <td>SUBJ</td> <td>< NP::[1] ></td> </tr> <tr> <td>COMPS</td> <td><></td> </tr> </table>	SUBJ	< NP::[1] >	COMPS	<>	
	SUBJ	< NP::[1] >						
COMPS	<>							
CONTENT	<table> <tr> <td>jít</td> <td></td> </tr> <tr> <td>KDO</td> <td>[1]</td> </tr> </table>	jít		KDO	[1]			
jít								
KDO	[1]							

DÁT	CATEGORY	HEAD	verb								
		VALENCE	<table> <tr> <td>SUBJ</td> <td>< NP::[1] ></td> </tr> <tr> <td>COMPS</td> <td>< NP::[2], NP::[3] ></td> </tr> </table>	SUBJ	< NP::[1] >	COMPS	< NP::[2], NP::[3] >				
	SUBJ	< NP::[1] >									
COMPS	< NP::[2], NP::[3] >										
CONTENT	<table> <tr> <td>dát</td> <td></td> </tr> <tr> <td>KDO</td> <td>[1]</td> </tr> <tr> <td>CO</td> <td>[2]</td> </tr> <tr> <td>KOMU</td> <td>[3]</td> </tr> </table>	dát		KDO	[1]	CO	[2]	KOMU	[3]		
dát											
KDO	[1]										
CO	[2]										
KOMU	[3]										

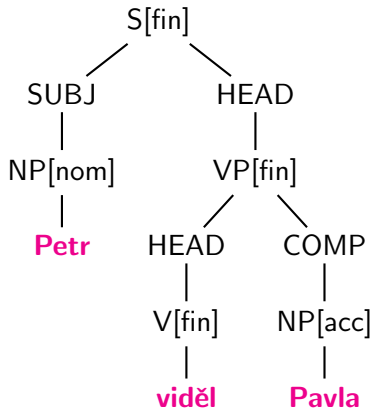
HPSG – fráze

reprezentace **fráží** – v HPSG obdoba reprezentace **slov**
 navíc příznak **DAUGHTERS** – struktura členů fráze



HPSG – fráze – pokrač.

pro snazší čtení popisů frází používáme **stromový zápis**:



ve skutečnosti se ovšem jedná o **příznakovou strukturu**, ne strom!

HPSG – dobře utvořené příznakové struktury

dobře utvořené příznakové struktury musí splňovat **omezení daná gramatikou**

příznaková struktura je **dobře utvořená** \Leftrightarrow :

- ▶ každý uzel splňuje **omezení geometrie příznaku**
- ▶ každá uzel vstupního slova splňuje **omezení** některé **lexikální položky**
- ▶ každý frázový uzel splňuje **frázová omezení** – *omezení přímé dominance* (immediate dominance, viz dále), *omezení hlavových příznaků* (head feature), *valenční omezení*, ...

omezení geometrie příznaku specifikují:

- ▶ s jakými **typy** se pracuje
- ▶ jaká je použitá **typová hierarchie** – který typ je podtypem jiného typu
- ▶ pro každý typ – jaké příznaky přísluší tomuto typu
- ▶ pro každý typ a každý příznak – jakých typů mohou být hodnoty tohoto příznaku

HPSG – deklarace typu

pro popis omezení geometrie příznaku se používají **typové deklarace**:

```
category: [HEAD: head, VALENCE: valence]
```

```
head      # příznaková struktura složená z příznakových struktur
```

```
  noun: [CASE: case]
```

```
  verb: [VFORM: vform, AUX: boolean, INV: boolean]
```

```
  prep: [PFORM: pform]
```

```
  ...
```

```
vform     # jednoduchý příznak, forma slovesa – možné hodnoty:
```

```
  fin     # určitý tvar slovesa
```

```
  inf     # neurčitý tvar slovesa – infinitive
```

```
  ...
```

```
case      # jednoduchý příznak, gramatický pád
```

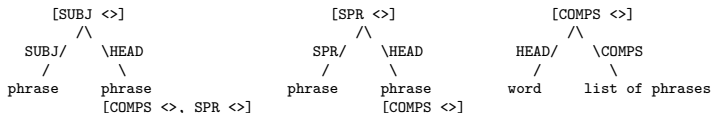
```
  nom     # 1. pád, nominativ
```

```
  acc     # 4. pád, akuzativ
```

```
  ...
```

HPSG – dobře utvořená slova a fráze

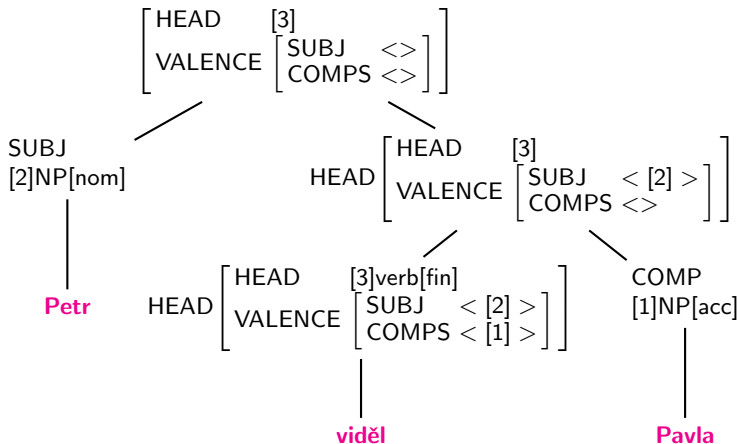
- ▶ každé vstupní **slovo** musí splňovat některou **lexikální položku**
- ▶ **fráze** musí splňovat **frázová omezení** (constraints):
 - **omezení přímé dominance** – každá fráze musí odpovídat jednomu ze schémat – schéma *head-subject*, schéma *head-specifier*, schéma *head-complement*, ...



- **omezení hlavových příznaků** – pro každou frázi, která má hlavu, musí být hlavové příznaky fráze shodné s hlavovými příznaky potomka, který je hlavou
- **valenční omezení** – pro každý z valenčních příznaků (SUBJECT, COMPLEMENTS, ...) – hodnota příznaku na hlavové frázi musí odpovídat hodnotě na potomku, který je hlavou, minus ty příznaky, které jsou splněny některým z nehlavových potomků

HPSG – dobře utvořené příznakové struktury

omezení ve větě 'Petr viděl Pavla.':



DEMO: [English Resource Grammar](http://www.delph-in.net/erg/) <http://www.delph-in.net/erg/>
 Enju <http://www.nactem.ac.uk/tsujii/enju/>

Syntaktický analyzátor SET

Syntactic Engineering Tool, autor Vojtěch Kovář

- ▶ důraz na **jednoduchost** v návrhu i v použití
- ▶ některé syntaktické jevy jsou lépe **rozpoznatelné** než jiné
- ▶ nejprve určíme **snadnější vztahy**, dále pokračujeme **složitějšími**

Principy:

- ▶ využití principů **parciální analýzy** pro analýzu úplnou
- ▶ pravidlový systém – množina **vzorků** (patterns)
- ▶ **pattern matching** – vyhledávání vzorků v textu

SET – jazyk pro definici pravidel

Každé **pravidlo** obsahuje dvě části – **šablonu** a **akce**

- ▶ **šablona** určuje, **co** se v textu má hledat
- ▶ **akce** určují, jaké **syntaktické vztahy** mají být vyznačeny
- ▶ a morfologické **shody**
- ▶ **pravděpodobnostní ohodnocení** nalezených vzorků – délka, pravděpodobnost pravidla

Příklady pravidel:

```
prep ... noun          AGREE 0 2 c MARK 2 DEP 0 PROB 500  
verb ... comma conj ... verb ... bound          MARK 2 7 <relclause>
```

SET – příklady pravidel

Podmínka pro **jedno slovo**:

```
(lemma world)
(word and|or|so)
(tag k[123].*c2)
```

Podmínka pro **více slov**:

```
noun ... noun2
```

```
$C1 (word and) $C2
MATCH $C1(tag) $C2(tag)
k1 k1
k2 k2
END
```


SET – příklady pravidel

Alias:

```
CLASS vpart (word by|bychom|byste|bych|bys)
```

```
CLASS noun (tag k1)
```

```
CLASS noun2 (tag k1c2)
```

Akce:

- ▶ **MARK** – vyznačuje závislosti a frázové prvky
- ▶ **DEP** – doplnění MARK, udává závislost
- ▶ **HEAD** – doplnění MARK, udává hlavu frázového prvku
- ▶ **AGREE** – požadavek na shodu (g/n/c)
- ▶ **PROB** – udává pravděpodobnostní váhu pravidla

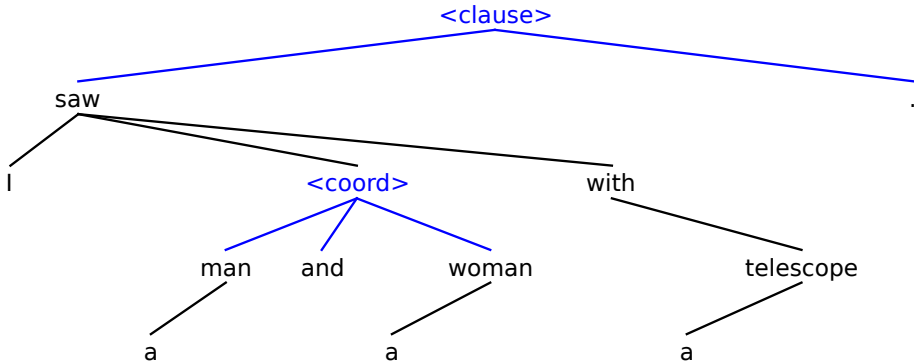
SET – výstup analýzy

hybridní stromy – kombinují **závislostní** a **složkové** prvky

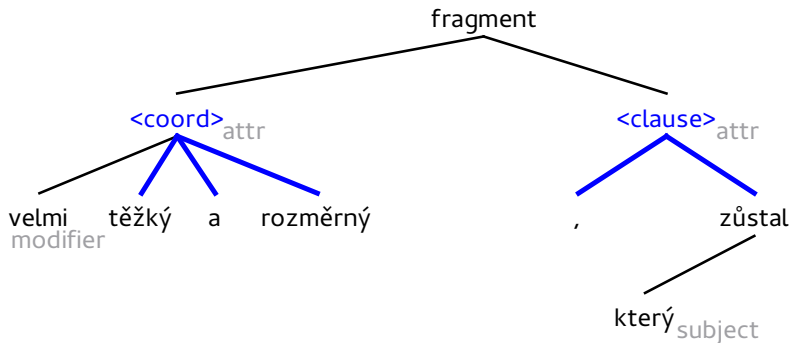
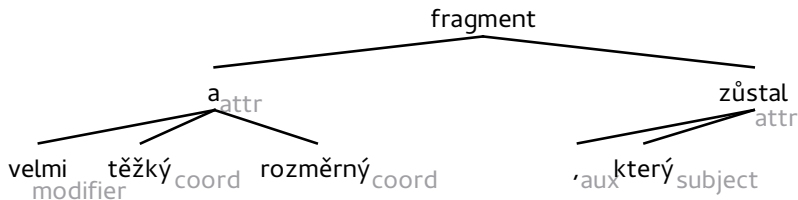
- ▶ **čitelnější** pro člověka
- ▶ rozlišování složkových a závislostních jevů je **výhodou** při analýze
- ▶ možnost **převodu** do čistě závislostního i čistě složkového formátu

Na výstupu analýzy je vždy **jediný strom**, možnost výpisu **všech nalezených vzorků** – zachycení možné víceznačnosti

Hybridní strom – příklad



Hybridní a závislostní strom



SET – implementace

Technické detaily

- ▶ implementace v jazyce **Python**
- ▶ **objektový model** věty, pravidel a syntaktických vztahů
- ▶ ucelený **soubor pravidel** pro analýzu syntaxe **češtiny**
- ▶ gramatiky pro **angličtinu**, **slovenštinu**
- ▶ specializované gramatiky pro **extrakce informací**, **opravy chyb** (interpunkce), ...
- ▶ 3000 řádků kódu, **70 pravidel**

Funkce:

- ▶ analýza **morfologicky označovaného textu**
- ▶ výstup ve formě různých typů **stromů**, **frází** a **kolokací**
- ▶ reprezentace **víceznačnosti**
- ▶ grafická **vizualizace** výstupu

SET – přesnost a rychlost

Rychlost:

- ▶ asymptoticky $O(R N^2 \log(R N^2))$
- ▶ v praxi 0.14 sekundy na větu

Přesnost závislostního výstupu (vzhledem k PDT, SET v0.3):

Testovací sada	Přesnost – průměr	Přesnost – medián
PDT e-test	76,14 %	78,26 %
BPT2000	83,02 %	87,50 %
PDT50	92,68 %	94,99 %

<http://nlp.fi.muni.cz/projekty/set/>

Metagramatika systému synt

3 formy (meta)gramatiky: [ukázka](#)

▶ metagramatika (G1)

- ▶ pravidla s kombinatorickými konstrukty + globální omezení pořadí
- ▶ akce (= gramatické testy + kontextové akce)
- ▶ česká lingvistická tradice – závislostní struktury, kontrola shody, pravidla pro pořadí slov, ...

▶ generovaná gramatika (G2)

- ▶ bezkontextová pravidla
- ▶ akce

▶ expandovaná gramatika (G3)

- ▶ jen bezkontextová pravidla

G1 - metagrammar	G2	G3
<pre> vol_list -> VOL /* muset a chtit */ &list_coord voi_list /* muset */ voi_list -> VOI /* budu muset a budu chtit */ &list_coord vbuvvoi_list /* budu muset */ vbuvvoi_list --> order(VBU, voi_list) /* musel jsem a chtel jsem */ &list_coord volvb12_list /* musel jsem */ volvb12_list --> order(vol_list, VB12) /* musel bych a chtel bych */ &list_coord volvbk_list /* musel bych */ volvbk_list --> order(vol_list, VBK) &list_coord_case prep /* bez */ prep -> PREP propagate_case(\$1) pn -> prep npn agree_case_and_propagate(\$1, \$2) depends(\$1, \$2) add_prep_ngroup(\$2) rule_schema(\$0, "lwt([awt(#1), try(#2)])") &list_coord pp /* z mesta */ pp -> pn /* castecne i z mesta */ pp -> part pn head(\$2) #/* z mesta nez z vesnice */ #pp -> pn NEZ pn # depends(\$2, \$1) # depends(\$2, \$3) </pre>	<pre> volvbk_list -> volvbk_listnl conjgconj v depends(\$2, \$1, \$3) head(\$2) volvbk_list -> volvbk_listnl /* musel bych */ volvbk_listnl -> vol_list intr VBK volvbk_listnl -> VBK intr vol_list prep -> prepnl conjgconj prep depends(\$2, \$1, \$3) head(\$2) agree_case_and_propagate(\$1, \$3) prep -> prepnl propagate_case(\$1) /* bez */ prepnl -> PREP propagate_case(\$1) pn -> prep npn agree_case_and_propagate(\$1, \$2) depends(\$1, \$2) add_prep_ngroup(\$2) rule_schema(\$0, "lwt([awt(#1), try(#2)])") pp -> ppnl conjgconj pp depends(\$2, \$1, \$3) head(\$2) pp -> ppnl /* z mesta */ ppnl -> pn /* castecne i z mesta */ ppnl -> part pn head(\$2) /* on ten (Petr je pekny ...) */ first_pron_group -> ON first_pron agree_case_number_gender_and_propagate head(\$2) head(\$1) /* ten (Petr je pekny ...) */ </pre>	<pre> volvb12_list -> volvb12_listnl volvb12_listnl -> vol_list intr VB12 volvb12_listnl -> VB12 intr vol_list volvbk_list -> volvbk_listnl conjgconj volv volvbk_list -> volvbk_listnl volvbk_listnl -> vol_list intr VBK volvbk_listnl -> VBK intr vol_list prep1 -> prepnl1 conjgconj prep1 prep2 -> prepnl2 conjgconj prep2 prep3 -> prepnl3 conjgconj prep3 prep4 -> prepnl4 conjgconj prep4 prep5 -> prepnl5 conjgconj prep5 prep6 -> prepnl6 conjgconj prep6 prep7 -> prepnl7 conjgconj prep7 prep1 -> prepnl1 prep2 -> prepnl2 prep3 -> prepnl3 prep4 -> prepnl4 prep5 -> prepnl5 prep6 -> prepnl6 prep7 -> prepnl7 prepnl1 -> PREP1 prepnl2 -> PREP2 prepnl3 -> PREP3 prepnl4 -> PREP4 prepnl5 -> PREP5 prepnl6 -> PREP6 prepnl7 -> PREP7 PREP1 -> PREP1SM PREP1 -> PREP1SI PREP1 -> PREP1SF PREP1 -> PREP1SN PREP1 -> PREP1PM PREP1 -> PREP1PI PREP1 -> PREP1PF PREP1 -> PREP1PN PREP2 -> PREP2SM PREP2 -> PREP2SI PREP2 -> PREP2SF PREP2 -> PREP2SN PREP2 -> PREP2PM PREP2 -> PREP2PI PREP2 -> PREP2PF PREP2 -> PREP2PN </pre>
<p style="text-align: right;">Rules: 1 / 345</p>	<p style="text-align: right;">Rules: 2 / 3102</p>	<p style="text-align: right;">Rules: 14 / 11556</p>
<p>Close Clicked Line: 763</p>	<p style="text-align: center;">File: /mnt/scsi-5/mp/projekty/grammar_workbench/synt/synt/grammars/synt.g1</p>	

Metagramatika – kombinatorické konstrukty

kombinatorické konstrukty se používají pro generování variant pořadí daným terminálů a neterminálů

hlavní kombinatorické konstrukty:

- ▶ **order()** generuje všechny možné permutace zadaných komponent
- ▶ **first()** argument musí být na prvním místě
- ▶ **rhs()** doplní všechny pravé strany svého argumentu

```
/* budu se ptát */
```

```
clause ==> order(VBU,R,VRI)
```

```
/* který ... */
```

```
relclause ==> first(relprongr) rhs(clause)
```

Metagramatika – typy pravidel

- ▶ \rightarrow normální CF pravidlo
- ▶ $\rightarrow\rightarrow$ vložit **intersegment** mezi každé dva prvky
- ▶ \Rightarrow + kontrola správného pořadí příklonek
- ▶ $\Rightarrow\Rightarrow$ intersegmenty na začátku a konci RHS, spojky, ...

ss \rightarrow conj clause

/* budu muset číst */

futmod $\rightarrow\rightarrow$ VBU VOI VI

/* byl bych býval */

cpredcondgr \Rightarrow VBL VBK VBLL

/* musím se ptát */

clause $\Rightarrow\Rightarrow$ VO R VRI

clause pravidla se zadávají pomocí **pravidlových vzorů**

Metagramatika – globální omezení pořadí

globální omezení pořadí zakazuje některé kombinace pořadí preterminálů

`%enclitic` – které preterminály jsou brány jako příklonky

`%order` – zajišťuje dodržení precedence zadaných preterminálů

```
/* jsem, bych, se */
```

```
%enclitic = (VB12, VBK, R)
```

```
/* byl — četl, ptal, musel */
```

```
%order VBL = {VL, VRL, VOL}
```

Metagramatika – generativní konstrukty

skupina výrazů **%list_*** – produkují nová pravidla pro seznamy (s oddělovači/bez oddělovačů, s různými testy na shody, ...)

```
/* (nesmim) zapomenout udelat - to forget to do */
```

```
%list_nocoord vi_list
```

```
vi_list -> VI
```

```
%list_coord_case np
```

```
%list_coord_case_number_gender left_modif
```

```
/* krasny velky pes a mala kocka - beautiful dog and small cat */
```

```
np -> left_modif np
```

koncovky ***_case**, ***_number_gender** and ***_case_number_gender** určují typ shody

Metagramatika – pravidlové vzory

pravidla pro slovesné skupiny – cca 40% všech pravidel metagramatiky
pravidlové vzory %group – definují časté skupiny konstrukcí v pravidlech

```
%group verbP={
  V:   verb_rule_schema($@,"(#1)")
        groupflag($1,"head"),
  VR R: verb_rule_schema($@,"(#1 #2)")
        groupflag($1,"head"),
}

%template clause ===> order(RHS)

/* ctu/ptam se - I am reading/I am asking */
clause %> group(verbP) vi_list
  verb_rule_schema($@,"#2")
  depends(getgroupflag($1,"head"), $2)
```

Metagramatika – pravidlové vzory – pokrač.

- ▶ předchozí příklad – skupina `verbP` = dvě skupiny preterminálů (`V` a `VR R`) s příslušnými akcemi
- ▶ při použití v `clause` vytvoří postupně dvě různé pravé strany
- ▶ `(get)groupflag` – odkaz na prvek uvnitř `%group`
- ▶ vzor celého pravidla – speciální pravidlová šipka `%>`
`%template` definuje vzor každého pravidla s `%>`

Metagramatika – úrovně pravidel

- ▶ používá se pro **ohodnocení** výstupních stromů pro jejich **třídění**
- ▶ doplněk trénování na **stromových korpusech** (6.000 vět)
- ▶ zadané **lingvistou** – specialistou na vývoj gramatiky
- ▶ **základní úroveň** – 0, **vyšší úrovně** – méně frekventované fenomény
- ▶ pravidla vyšších úrovní mohou být v průběhu analýzy **zapnuté/vypnuté**

```
3:np -> adj_group  
  propagate_case_number_gender($1)
```

Gramatika G2 – kontextové akce

- ▶ gramatické **testy na shody** – pád, rod, číslo
- ▶ **testy na zanoření vedlejších vět** – test_comma
- ▶ akce pro specifikaci **závislostních hran**
- ▶ akce **typové kontroly** logických konstrukcí

```
np -> adj_group np
```

```
rule_schema($@, "lwtx(awtx(#1) and awtx(#2))")
```

```
rule_schema($@, "lwtx([[awt(#1),#2],x])")
```

rule_schema – schéma pro tvorbu logické konstrukce ze subkonstrukcí

projdou jenom kombinace, které **typově vyhovují** danému schématu

Expandovaná gramatika G3

- ▶ překlad testů na shody do CF pravidel
- ▶ v češtině – 7 gramatických pádů, dvě čísla a 4 rody → 56 možných variant pro plnou shodu mezi dvěma prvky

počty pravidel

metagramatika G1	253
gramatika G2	3091
expandovaná gramatika G3	11530

Výstupy syntaktické analýzy

synt nabízí více možností zpracování výsledných struktur:

- ▶ **syntaktické stromy** (varianty: technická/lingvistická, uspořádané/neuspořádané) ▶ ukázka
- ▶ struktura **chart** – komprimovaný les všech stromů ▶ ukázka
- ▶ **závislostní graf** – graf všech závislostí vytvořených akcemi ▶ ukázka
- ▶ seznamy **frází** v dané větě, získané přímo ze struktury **chart** ▶ ukázka
- ▶ částečné **zjednoznačnění morfologických značek** na vstupu ▶ ukázka
- ▶ převod na **logické konstrukce TIL** ▶ ukázka

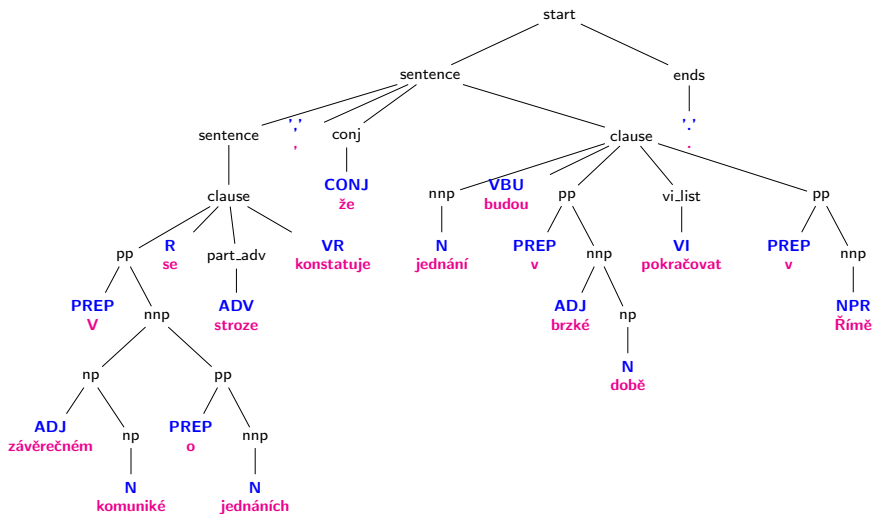
manuál ke **GDW** – Grammar Development Workbench

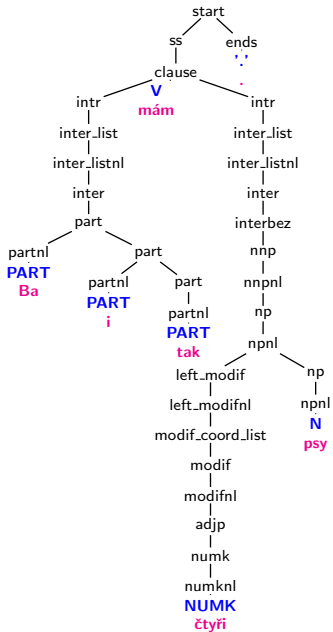
http://nlp.fi.muni.cz/projekty/grammar_workbench/manual/

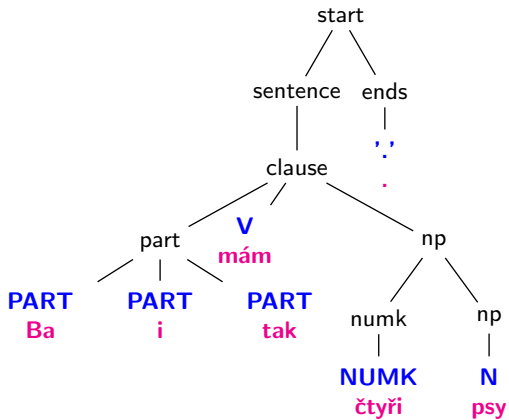
DEMO: **wwwsynt** – webové rozhraní k syntu

<http://nlp.fi.muni.cz/projekty/wwwsynt/>

V závěrečném komuniké o jednáních se stroze konstatuje, že jednání budou v brzké době pokračovat v Římě.





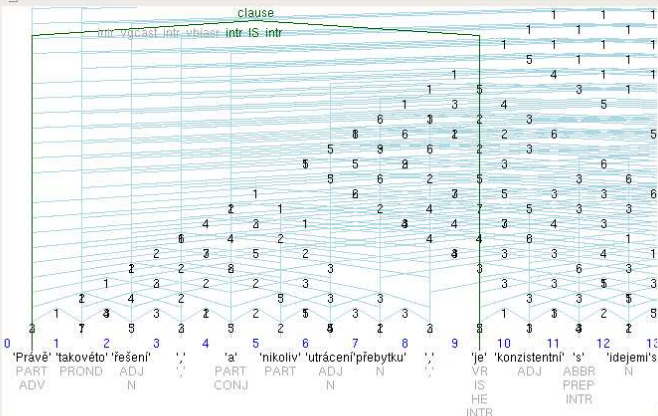


< > 0 ... 16 7346 / 7346

< -> Fix Edge

418 0 12 clause -> intr vgca
 419 0 14 clause -> intr vgca
 420 0 13 clause -> intr vgca
 421 0 15 clause -> intr vgca
 422 0 10 clause -> intr vgca
 423 0 12 clause -> intr vbias
 424 0 14 clause -> intr vbias
 425 0 13 clause -> intr vbias
 426 0 15 clause -> intr vbias
 427 0 11 clause -> intr vbias

```
->422: (5980,505) 0 10 clause -> intr vgoast intr vbias intr { IS } intr :
5980: (5981,7262) 9 10 clause -> intr vgoast intr vbias intr { IS } intr :
505: (-1,47)(-1,506) 0 9 intr -> { inter_list } :
```

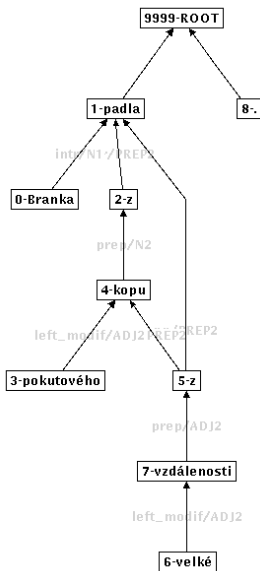


INFO: Closed edges ranges displayed.

Select 0 / 649

left_modifnl -> modif_coord
 modif -> modifnl
 modif -> modifnl conjconj m
 modif_coord_list -> modif
 modifnl -> pre_adj adjp
 modifnl -> adjp
 nnp -> nnpnl conjconj nnp
 nnp -> nnpnl
 nnpnl -> np
 np -> nnpnl conjconj np

Branka padla z pokutového kopu z velké vzdálenosti.



np: *Tyto normy se však odlišují nejen v rámci různých národů a států, ale i v rámci sociálních skupin, a tak považují dřívější pojetí za dosti široké a nedostačující.*

[0-2) Tyto normy

[2-3) se

[6-12) v rámci různých národů a států

[15-19) v rámci sociálních skupin

[23-30) dřívější pojetí za dosti široké a nedostačující

vp: *Kdybych to byl býval věděl, byl bych sem nechodil.*

[0-5): byl býval věděl

[6-10): byl bych nechodil

clause: *Muž, který stojí u cesty, vede kolo.*

[0-9): Muž , , vede kolo

[2-6): který stojí u cesty

slovo	před	po
Na	k7{c4, c6}	k7c6
krásné	k2eA{gFnPcld1, gFnPc4d1, gFnPc5d1, gFnSc2d1, gFnSc3d1, gFnSc6d1, glnPcld1, glnPc4d1, glnPc5d1, glnScld1wH, glnSc4d1wH, glnSc5d1wH, gMnPc4d1, gMnScld1wH, gMnSc5d1wH, gNnScld1, gNnSc4d1, gNnSc5d1}	k2eAgFnSc6d1
dlouhé	k2eA{gFnPcld1, gFnPc4d1, gFnPc5d1, gFnSc2d1, gFnSc3d1, gFnSc6d1, glnPcld1, glnPc4d1, glnPc5d1, glnScld1wH, glnSc4d1wH, glnSc5d1wH, gMnPc4d1, gMnScld1wH, gMnSc5d1wH, gNnScld1, gNnSc4d1, gNnSc5d1}	k2eAgFnSc6d1
ulici	klgFnSc3, klgFnSc4, klgFnSc6	klgFnSc6
stálo	k5eAalmAgNnSaIrD	kSeApNnStMmPaI
moderní	k2eA{gFnPcld1, gFnPc4d1, gFnPc5d1, gFnScld1, gFnSc2d1, gFnSc3d1, gFnSc4d1, gFnSc5d1, gFnSc6d1, gFnSc7d1, glnPcld1, glnPc4d1, glnPc5d1, glnScld1, glnSc4d1, glnSc5d1, gMnPcld1, gMnPc4d1, gMnPc5d1, gMnScld1, gMnSc5d1, gNnPcld1, gNnPc4d1, gNnPc5d1, gNnScld1, gNnSc4d1, gNnSc5d1}	k2eAgNnScld1, k2eAgNnSc4d1, k2eAgNnSc5d1
nablýskané	k2eA{gFnPcld1rD, gFnPc4d1rD, gFnPc5d1rD, gFnSc2d1rD, gFnSc3d1rD, gFnSc6d1rD, glnPcld1rD, glnPc4d1rD, glnPc5d1rD, glnScld1wHrD, glnSc4d1wHrD, glnSc5d1wHrD, gMnPc4d1rD, gMnScld1wHrD, gMnSc5d1wHrD, gNnScld1rD, gNnSc4d1rD, gNnSc5d1rD}	k2eAgNnScld1, k2eAgNnSc4d1, k2eAgNnSc5d1
auto	klgNnScI, klgNnSc4, klgNnSc5	klgNnScI, klgNnSc4, klgNnSc5

System synt – příklad logické analýzy

vyhodnocení `rule_schema` pro `np` 'pečené kuře'

```
4, 6, -npl -> . left_modif np .: k1gNnSc145
```

```
agree_case_number_gender_and_propagate OK
```

```
rule_schema: 2 nterms, 'lwtx(awtx(#1) and awtx(#2))'
```

```
And constrs, Abstr and Exi vars are just gathered
```

```
1 (1x1) constructions:
```

$$\lambda w_2 \lambda t_3 \lambda x_4 ([\text{pečený}_{w_2 t_3, x_4}] \wedge [\text{kuře}_{w_2 t_3, x_4}]) \dots (ol)_{\tau\omega}$$

```
And constrs: none added
```

```
Exi vars: none added
```

Systém synt – příklad logické analýzy – pokrač.

vyhodnocení **verb_rule_schema** pro celou **clause**

verb_rule_schema: 3 groups

no acceptable subject found: supplying an inexplicit one

inexplicit subject: k3xPgMnSc1,k3xPgInSc1: $On \dots \iota$

Clause valency list: jíst <v>#1:(1)hA-#2:(2)hPTc1, ...

Verb valency list: jíst <v>#2:hH-#1:hPTc4ti

Matched valency list: jíst <v>#2:(1)hH-#1:(2)hPTc4ti

time span: $\lambda t_{12} \mathbf{dnes}_{tt_{12}} \dots (o\tau)$

frequency: $\mathbf{Onc} \dots ((o(o\tau))\pi)_{\omega}$

verbal object: $x_{15} \dots (o(o\pi)(o\pi))$

present tense clause:

$$\lambda w_{17} \lambda t_{18} (\exists i_{10}) (\exists x_{15}) (\exists i_{16}) ([\mathbf{Does}_{w_{17}t_{18}}, On, [\mathbf{Imp}_{w_{17}}, x_{15}]] \wedge [\mathbf{večeře}_{w_{17}t_{18}}, i_{10}] \wedge$$

$$[\mathbf{pečený}_{w_{17}t_{18}}, i_{16}] \wedge [\mathbf{kuře}_{w_{17}t_{18}}, i_{16}] \wedge x_{15} =$$

$$[\mathbf{jíst}, i_{16}]_{w_{17}} \wedge [[\mathbf{k}_{w_{17}t_{18}}, i_{10}]_{w_{17}}, x_{15}]) \dots \pi$$

clause:

$$\lambda w_{19} \lambda t_{20} [\mathbf{P}_{t_{20}}, [\mathbf{Onc}_{w_{19}}, \lambda w_{17} \lambda t_{18} (\exists i_{10}) (\exists x_{15}) (\exists i_{16}) ([\mathbf{Does}_{w_{17}t_{18}}, On, [\mathbf{Imp}_{w_{17}}, x_{15}]] \wedge$$

$$[\mathbf{večeře}_{w_{17}t_{18}}, i_{10}] \wedge [\mathbf{pečený}_{w_{17}t_{18}}, i_{16}] \wedge [\mathbf{kuře}_{w_{17}t_{18}}, i_{16}] \wedge x_{15} =$$

$$[\mathbf{jíst}, i_{16}]_{w_{17}} \wedge [[\mathbf{k}_{w_{17}t_{18}}, i_{10}]_{w_{17}}, x_{15}])], \lambda t_{12} \mathbf{dnes}_{tt_{12}}] \dots \pi$$