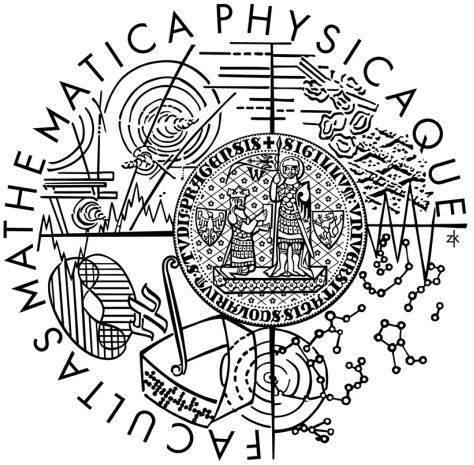
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Ondřej Kučera

Pražský závislostní korpus jako cvičebnice jazyka českého

Ústav formální a aplikované lingvistiky

Vedoucí diplomové práce:
Mgr. Barbora Vidová Hladká, PhD.

Studijní program: Informatika, matematická lingvistika

Děkuji vedoucí své diplomové práce Mgr. Barboře Vidové Hladké, PhD. za cenné rady udělované v celém průběhu psaní práce.

Prohlašuji, že jsem svou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 15. 12. 2005

Ondřej Kučera

Obsah

Název práce: Pražský závislostní korpus jako cvičebnice jazyka českého

Autor: Ondřej Kučera

Katedra (ústav): Ústav formální a aplikované lingvistiky

Vedoucí diplomové práce: Mgr. Barbora Vidová Hladká, PhD.

e-mail vedoucího: hladka@ufal.mff.cuni.cz

Abstrakt: Pražský závislostní korpus (PDT) patří mezi nejvýznamnější jazykové korpusy na světě. Cílem této práce je představit softwarový systém, který nad daty PDT tvoří cvičebnici českého jazyka. Procvičování probíhá ve dvou oblastech: tvarosloví (určování slovních druhů a jejich morfologických kategorií) a větný rozbor (určování větných členů a závislostí mezi nimi). Vzhledem k odlišnostem mezi akademickými rozbory vět a rozbory tak, jak jsou vyučovány ve školách, však nelze data PDT použít zcela přímočaře. Mnoho vět je potřeba z dat úplně vyřadit, na ostatních je nutné provést množství transformací, které převedou původní reprezentaci do tvaru, na který jsou žáci zvyklí ze školy.

Klíčová slova: Pražský závislostní korpus, cvičebnice češtiny, zpracování přirozeného jazyka, větný rozbor

Title: Prague Dependency Treebank as an excercise book of Czech language

Author: Ondřej Kučera

Department: Institute of Formal and Applied Linguistics

Supervisor: Mgr. Barbora Vidová Hladká, PhD.

Supervisor's e-mail address: hladka@ufal.mff.cuni.cz

Abstract: Prague Dependency Treebank (PDT) is one of the top language corpora in the world. The aim of this work is to introduce a software system that builds an exercise book of Czech using the data of PDT. Two kinds of exercises are provided: morphology (selecting correct parts of speech and their morphological cathegories) and sentence parsing (selecting analytical functions and dependencies between them). The PDT data cannot be used directly though, because of the differences between the academic approach in sentence parsing and the approach that is used in schools. Some of the sentences have to be discarded completely, several transformations have to be applied to the others in order to convert the original representation to the form to which the students are used to from school.

Keywords: Prague Dependency Treebank, exercise book of Czech, natural language processing, sentence parsing

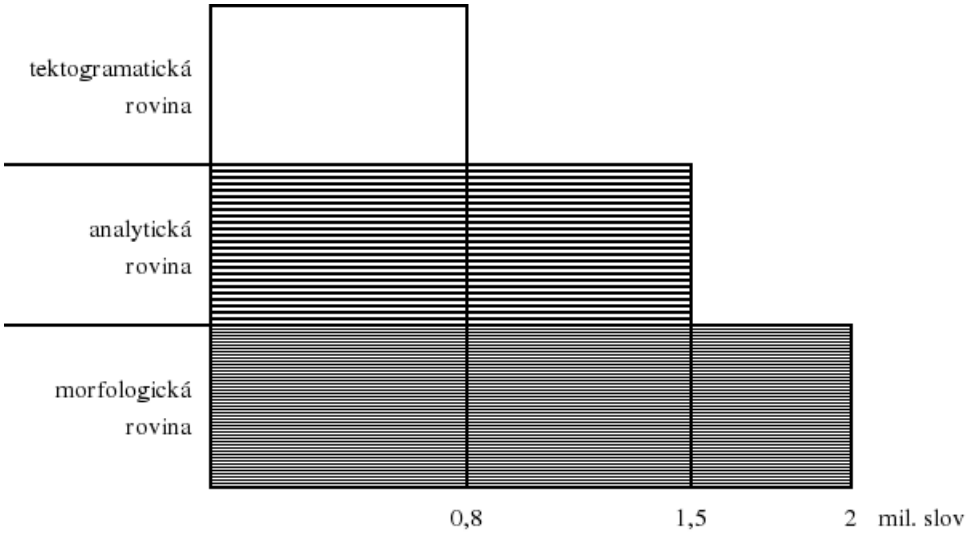
Kapitola 1

Úvod

Jako každá jiná vědní disciplína i lingvistika se dělí na řadu podoborů. V této práci pro nás nejdůležitějším z nich bude lingvistika korpusová. Samotná myšlenka jazykového korpusu, tedy nashromážděného množství textů toho kterého jazyka, není nikterak nová. Teprve příchod počítačů však posunul korpusovou lingvistiku na dnešní úroveň. Díky nim je možné nashromáždit, ale především rychle zpracovávat řádově mnohem větší množství dat, než by kdy bylo v lidských silách. Vzniklo tak další nové odvětví, počítačová (či komputační) lingvistika.

Na běžný elektronický korpus můžeme pohlížet jako na řetězec po sobě jdoucích slov (tvořících jednotlivé věty). Můžeme v něm velice snadno například hledat výskyty určitých slovních tvarů, porovnávat jejich počet nebo třeba zjišťovat průměrný počet slov ve větách. To vše můžeme provádět s korpusem prakticky libovolného jazyka, dokonce aniž bychom měli jeho zásadnější znalosti. Pro náročnější úlohy, jakými může být třeba strojový překlad mezi dvěma jazyky nebo systém zodpovídání dotazů, však potřebujeme hlubší analýzu korpusových dat. Tak například chceme-li v korpusu nalézt všechny výskyty všech tvarů slova *kočka*, musíme znát morfologické údaje o každém slovu v korpusu. Lze sice namítnout, že můžeme vyjmenovat jednotlivé tvary sami a pak hledat libovolný z nich, avšak tato strategie již selže v případě, kdy budeme hledat tvary podstatného jména *hnát*, protože bez morfologických informací obsažených přímo v datech nedokážeme posoudit, zda nalezený výskyt neodpovídá *slovesu* hnát. Můžeme však jít ještě dál a v korpusu uchovávat také třeba údaje o syntaktickém rozboru věty. Korpus obohacený o takováto metadata se nazývá korpusem *anotovaným*.

Pro český jazyk takový anotovaný korpus existuje – je jím *Pražský závislostní korpus* (Prague Dependency Treebank, dále často jen PDT, viz [1])[1]. Proč pražský a závislostní? Pražský pochopitelně proto, že vznikl v Praze a navazuje na tradici pražské lingvistické školy. Závislostní pak znamená, že z hlediska syntaktického je zvolen závislostní přístup, kdy za hlavní člen věty je považován predikát (nejčastěji sloveso), který je rozvíjen dalšími, závislými členy (které mohou být rovněž rozvíjeny). Pražský závislostní korpus je anotován na třech úrovních: na rovině morfologické (určení lemmat, slovních druhů a gramatických kategorií, jako jsou rod, číslo, pád, ...), analytické (syntaktické informace – analytická funkce, závislosti jednotlivých uzlů) a tektogramatické (rozbor sémantiky, významu). Anotace probíhaly od nejjednoduššího ke složitějšímu, tedy od morfologie k sémantice, čemuž odpovídá i množství označkovaných slov na jednotlivých rovinách. Slov s metadaty na všech třech rovinách je 0,8 mil., slov anotovaných morfologicky a analyticky 1,5 mil. a konečně na morfologické rovině je označkováno celkem 2,0 mil. slov (viz obrázek 1.1). Svými vlastnostmi se PDT řadí k předním světovým korpusům.



Obrázek 1.1: Rozložení počtu anotovaných slov v PDT na jednotlivých rovinách

Pražský závislostní korpus zpřístupňuje nejenom nové možnosti ověřování dřívějších lingvistických teorií, ale především umožňuje vytváření (a

[1]Vydání druhé verze PDT se připravuje na začátek roku 2006.

rovněž ověřování) teorií nových, zvláště v oblasti statistických metod a metod strojového učení. Cílem této diplomové práce je vytvořit počítačový systém, jenž bude využívat dat z PDT k sestavování úloh k procvičování tvarosloví a větných rozborů. Pokud je nám známo, tak u nás ani ve světě neexistuje podobný projekt, který by zpřístupňoval myšlenku jazykového korpusu školním dětem, klademe si za cíl rovněž popularizovat PDT jako akademický produkt mezi širokou veřejností.

Kapitola 2

Existující elektronické cvičebnice

2.1 Motivace

Dnešní žáci základních a středních škol běžně počítače používají. Hrají hry, surfují po Internetu, chatují s kamarády, píšou si deník nebo malují. Snadno se nabízí otázka, proč by se prostřednictvím počítačů nemohli i vzdělávat, konkrétně v našem případě proč by nemohli určovat morfologické kategorie slov či rozebírat větu a označovat větné členy. Pochopitelně nelze očekávat, že by snad děti byly z této možnosti nadšené tolik jako z těch předchozích jmenovaných. Na druhou stranu však gramatiku procvičovat tak jako tak potřebují a jak doufáme, pro mnohé z nich to bude takto snazší a třeba i zábavnější než při použití tradičních tištěných cvičebnic.

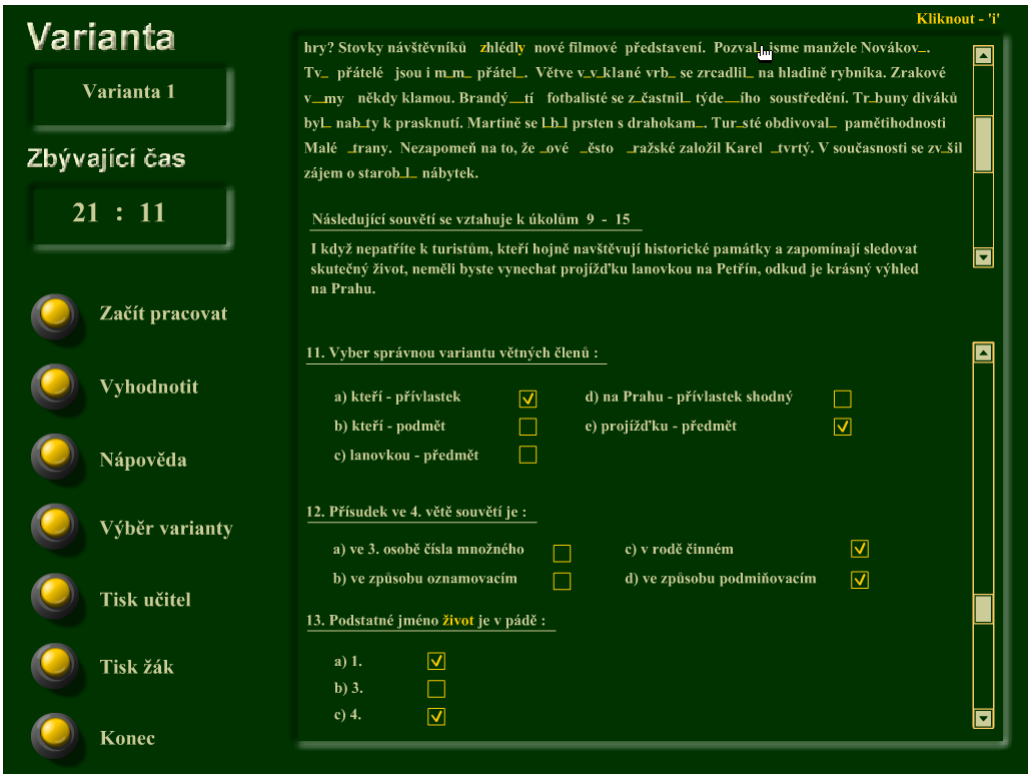
Samozřejmě řada elektronických učebnic a výukových programů již existuje, v tomto směru nejsme první, snažili jsme se proto rovněž vyzkoušet si dostupné produkty a inspirovat se jejich přednostmi, nebo se naopak poučit z jejich chyb. Tato kapitola bude stručně popisovat některé programy, s nimiž jsme se seznámili.

2.2 Český jazyk, přijímací zkoušky na SŠ

Program [9] obsahuje 40 komplexních cvičení skladbou odpovídajících pololetním srovnávacím testům na základních školách či právě přijímacím

testům středních škol. Cvičení obsahují celou řadu úkolů – doplňování i/y, s/z, čárek, určení správného typu odvození slov, objevují se v nich i otázky z literatury.

Nás pochopitelně nejvíce zajímalo řešení morfologie a syntaxe. Jak ukazuje obrázek 2.1, v tomto směru je pouze vybráno několik slov, u nichž uživatel určuje některé jejich charakteristiky.



Obrázek 2.1: Český jazyk, přijímací zkoušky na SŠ, ukázka cvičení

Ovládání programu je zaměřeno na myš, přesněji řečeno klávesnice se nedá použít vůbec, i určování i/y apod. probíhá klikáním myší. Jediná klávesa, která funguje, je Esc, ta způsobí, že program opustí režim plné obrazovky, do kterého se poté již nedá znovu přepnout. Zajímavé je, že i v případech, kdy uživatel musí evidentně určovat jedinou správnou variantu z několika možností, autoři programu volí výběr pomocí checkboxů a umožňují tak uživateli zaškrtnout, že například podstatné jméno se nachází hned v několika pádech (viz obrázek 2.1, otázka č. 13). Všechna cvi-

čení se dají rovněž vytisknout, a to jak v původní, tak i ve vyřešené podobě, tudíž se pak dají použít jako tradiční cvičebnice s klíčem. Zaměření programu je na žáky závěrečných ročníků základních škol.

2.3 TS Český jazyk 2 – jazykové rozbory

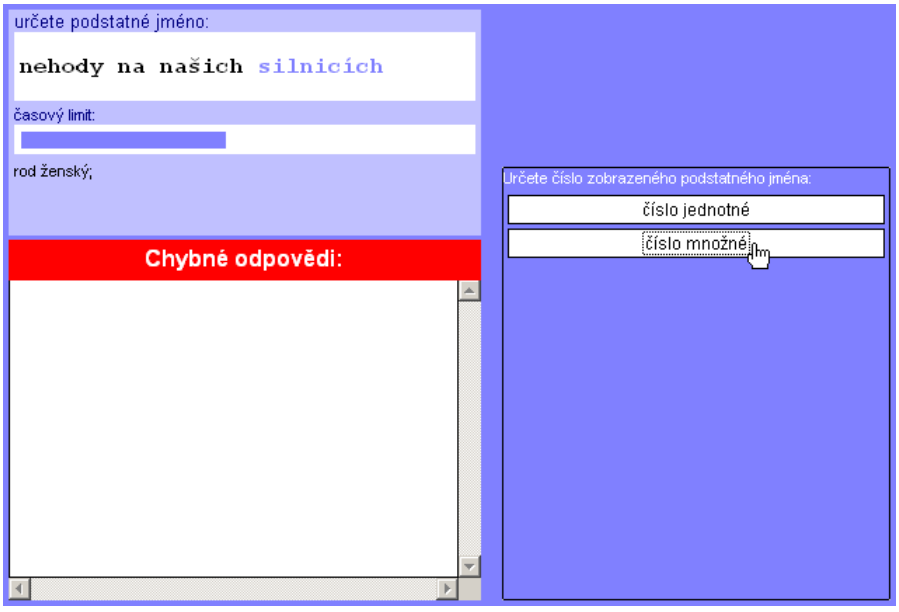
V programu [12] jsou cvičení rozdělena do osmi oblastí: slovesa, přídavná jména, podstatná jména, druhy vět vedlejších, větné rozbory, čárka ve větě jednoduché, větné členy, čárka v souvětí (viz obrázek 2.2). Samotná cvičení pak probíhají sérií otázek, na které uživatel musí v daném pořadí odpovědět během časového limitu. Například tedy zvolí-li uživatel úlohu z okruhu podstatných jmen, budou mu zobrazovány krátké výrazy a v nich bude vyznačeno podstatné jméno, u něhož uživatel postupně určí rod, číslo, pád a vzor (viz obrázek 2.3). Obdobným způsobem probíhají i cvičení z oblasti větných členů. U úloh z větných rozborů je uživateli prezentována celá (většinou jednodušší) věta, u níž má za úkol určit jednotlivé větné členy, zde si uživatel může určit pořadí, v němž bude větu zpracovávat (viz obrázek 2.4).

Ačkoliv je ovládání programu rovněž cíleno především na myš, pokud u něj uživatel chce strávit trochu více času, brzy přijde na to, že je efektivnější alespoň v rámci samotných cvičení používat kurzorových šipek a Enteru. Předností programu jsou (volitelné) namluvené komentáře, které u každé otázky zdůvodňují její správnou odpověď, aplikace tak nejen procvičuje, ale zároveň i vyučuje. Jednotlivé úlohy lze tisknout, zde je možno vybírat si konkrétní věty či jejich části, které budou vytištěny, nebo nechat provést náhodný výběr (na rozdíl od práce s cvičeními přímo na obrazovce, tam se o náhodný výběr jedná vždy). Pro nás nejzajímavější část, větné rozbory, obsahuje asi 400 vět o maximální délce přibližně okolo deseti slov.

2.4 Didakta Český jazyk 1

Program [8] poskytuje procvičování v oblasti větné skladby. Úlohy jsou rozděleny do pěti skupin: větné členy základní, větné členy rozvíjející, jednoduché věty, věty hlavní a vedlejší, souvětí (viz obrázek 2.5). U cvičení

Obrázek 2.2: TS Český jazyk 2, výběr cvičení

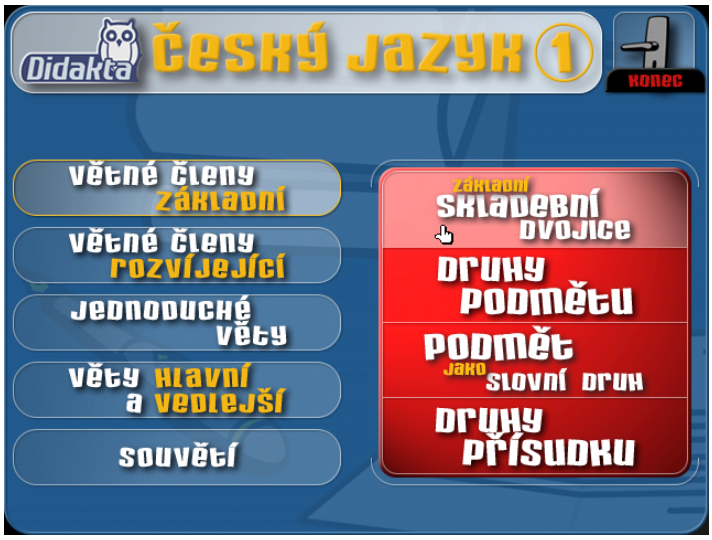


Obrázek 2.3: TS Český jazyk 2, podstatná jména

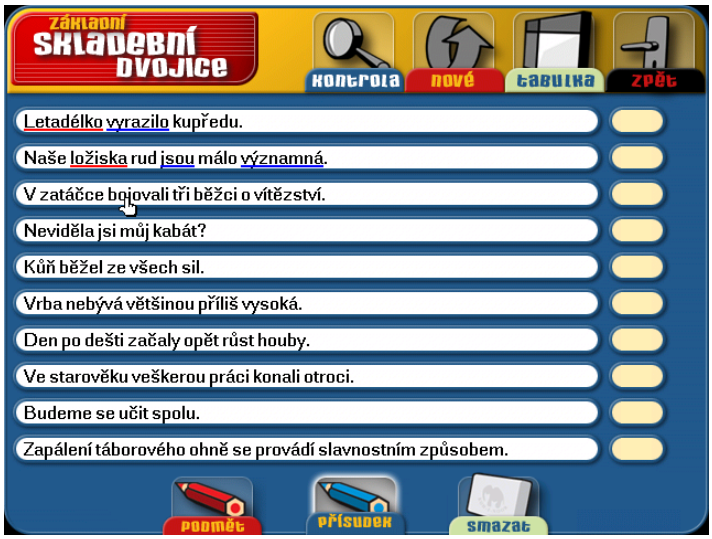
Obrázek 2.4: TS Český jazyk 2, větné rozbory

z určování základní skladební dvojice uživatel vždy zvolí buď červenou tužku pro podmět, nebo modrou pro přísudek a poté klikne na slovo tomu větnému členu odpovídající (viz obrázek 2.6).

Podle autorů je aplikace zaměřena na žáky vyššího stupně základních škol. Její vzhled a způsob ovládání by však byly vhodné spíše pro snad ještě mladší děti. Především u ovládání byla totiž zjevná snaha učinit jej takovým, aby uživatel měl co nejmenší pocit, že sedí u něčeho neznámého a možná trochu tajemného, jako je počítač. To ovšem paradoxně vede k tomu, že zkušenější uživatel chvílemi marně přemýšlí, co by měl vlastně udělat dál. Netroufáme si posoudit, zda toto cílová skupina žáků vnímá spíše positivně, nebo naopak negativně. Při spuštění navíc program přepne obrazovku do rozlišení 640×480 pixelů, což zejména na LCD monitorech může vést k nehezkému zobrazení.



Obrázek 2.5: Didakta Český jazyk 1, výběr cvičení



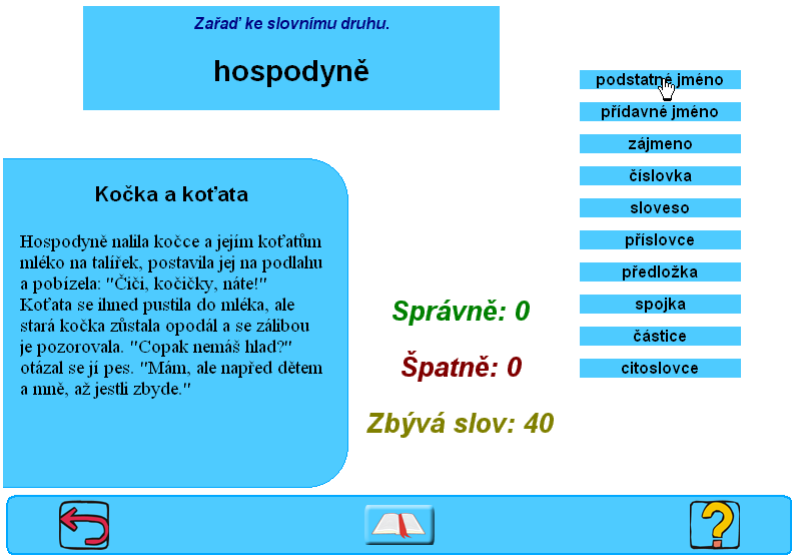
Obrázek 2.6: Didakta Český jazyk 1, určování základní skladební dvojice

2.5 PON Škola – Český jazyk

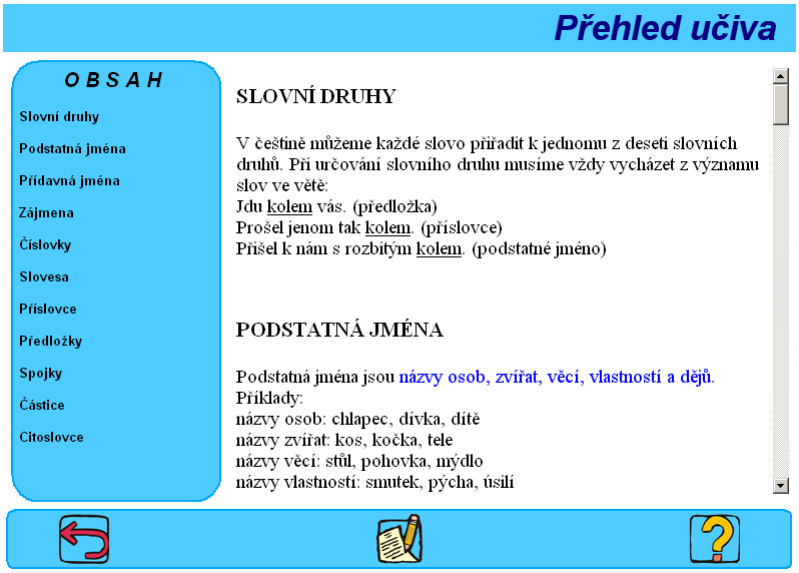
I u programu [14] se cvičení volí z několika oblastí, jak je vidět na obrázku 2.7. Pro nás bylo nejpodstatnější procvičování slovních druhů. Uživateli je zde zobrazen krátký text, z něhož jsou vybírána některá slova, u kterých uživatel určuje slovní druh (viz obrázek 2.8). V případě podstatných jmen a sloves lze navíc zvolit cvičení, kdy je uživatel zkoušen z morfologických kategorií těchto slovních druhů.

Obrázek 2.7: PON Škola – Český jazyk, výběr cvičení

Ovládání aplikace je opět možné jedině pomocí myši. To se jako poněkud nešikovné jeví obzvlášť ve chvíli, kdy během cvičení musí uživatel po každé (ať už správné, nebo špatné) odpovědi dojet myší nad tlačítko „Pokračovat“, stisknout jej a zase se vrátit do oblasti, ve které bude vybírat následující odpověď. Zajímavou vlastností programu je, že si uživatel může kdykoliv během procvičování nechat zobrazit přehled učiva (viz obrázek 2.9).



Obrázek 2.8: PON Škola – Český jazyk, určování slovních druhů



Obrázek 2.9: PON Škola – Český jazyk, přehled učiva

2.6 Srovnání s naší představou

Ačkoliv všechny programy, které jsme měli možnost vyzkoušet, nějakým způsobem umožňují procvičovat tvarosloví a syntax, ani jeden z nich se nepřibližuje našemu cíli – vzít větu a komplexně ji z těchto dvou hledisek rozebrat. Na druhou stranu tyto programy obsahují celou řadu jiných cvičení, kterým my se věnovat nechceme a ani nemůžeme. Především však žádný z nich na úrovni syntaktické nejde dál než k určení některých (popřípadě všech) větných členů ve větě, uživateli nejsou závislosti mezi jednotlivými větnými členy ani zobrazeny (například se správným řešením), natož aby mu bylo umožněno si označování těchto závislostí procvičit. V tomto směru můžeme rozhodně považovat výsledky naší práce za jedinečné.

Z hlediska ovládání nelze autorům zmíněných produktů upřít snahu vytvořit pokud možno intuitivní prostředí (především pro počítačově nezkušené). Někdy ovšem i za cenu toho, že pokročilejší uživatel je chvílemi mírně zmaten, pokud očekává chování obdobné běžným aplikacím, které zná. My jsme se oproti tomu rozhodli zvolit poněkud jiný přístup, chtěli jsme, aby naše cvičebnice naopak co nejvíce vypadala jako běžný program pro MS Windows[1]. Pokročilejší uživatel se pak snadněji vpraví do jejího ovládání a začátečník s tím sice může zpočátku mít trochu potíže, na druhou stranu však znalosti, které tak získá, upotřebí i později.

[1]MS Windows byly naší hlavní cílovou platformou, ovšem cvičebnice je přenositelná i na jiné operační systémy.

Kapitola 3

Budování cvičebnice

Chceme-li vytvořit cvičebnici češtiny (nebo obecně i jiného jazyka), můžeme postupovat dvěma způsoby. Zaprvé si můžeme všechnu práci udělat sami ručně. Věty si buď vymyslíme nebo je odněkud opíšeme (případně zkombinujeme obojí) a jednu po druhé je zpracujeme, určíme všechny slovní druhy, jejich mluvnické kategorie, větné členy a závislosti a cvičebnice je hotová. Tento přístup má hned několik nevýhod. Je pro autora nesmírně pracný, rovněž je dost náchylný k chybám. Nejspíš se nepodaří dát dohromady bohatší výběr vět než několik desítek (možná stovek), ale především je vysoce pravděpodobné, že zvolené věty nebudou příliš dobře reflektovat skutečné používání jazyka – patrně budou v průměru jednodušší a kratší. Výhodou tohoto řešení je, že je lze použít prakticky vždy.

Alternativně se můžeme pokusit sestavit cvičebnici automaticky (nebo možná přesněji poloautomaticky), ovšem za předpokladu, že máme k dispozici anotovaný korpus. Tento postup odstraňuje nevýhody předešlého – nejtěžší práce je již hotová, korpus existuje a je označkován. Chyby se v něm sice zajisté rovněž vyskytují, ale pravděpodobně v podstatně nižším rozsahu. Anotování korpusových vět totiž obvykle provádí více lidí najednou a šance, že se anotátoři shodnou na chybném řešení, je relativně malá. Hlavně však je-li korpus dobře sestaven, aby odrážel současný stav jazyka, bude tak činit i výsledná cvičebnice, stejně jako její velikost bude přímo úměrná velikosti celého korpusu.

V naší práci jsme se vydali právě touto druhou cestou. Jako anotovaný korpus jsme použili *Pražský závislostní korpus*. Jeho využití však nemohlo

být úplně přímočaré, bylo potřeba provést řadu úprav, o čemž budeme pojednávat v následujícím textu. Nejprve se zastavíme ještě u PDT samotného (kapitola 4), potom budeme hovořit o filtrování vět, které nebylo možné do cvičebnice zařadit (kapitola 5), a nakonec (kapitola 6) rozebereme, jaké je potřeba udělat transformace, abychom z rozborů vět analytické roviny PDT získali rozbory odpovídající školní výuce.

Kapitola 4

Pražský závislostní korpus

4.1 Logická struktura

Mnohé informace o *Pražském závislostním korpusu* jsme již zmínili v závěru kapitoly 1, zde zopakujeme ty nejdůležitější charakteristiky. PDT je anotován na třech rovinách: morfologické, analytické a tektogramatické (sémantické). Značkování probíhalo postupně od morfologické roviny, nejvíce, 2 mil., slov je tedy anotováno na ní, z nich 1,5 mil. i na analytické rovině a z nich 800 tisíc rovněž na rovině tektogramatické. PDT je navržen tak, aby na těchto rovinách bylo možné označkovat prakticky libovolnou větu, to má pochopitelně za následek, že pravidla anotování jsou poměrně komplikovaná. Ke každé rovině proto existuje anotační manuál, z těchto tří příruček ([3, 2, 5]) jsme při sestavování cvičebnice vycházeli.

4.2 Fyzická struktura

Vnitřním formátem PDT je *Prague Markup Language* (PML)[1], formát založený na jazyce XML. V něm jsou odděleny anotace na jednotlivých rovinách od samotného textu, existuje tedy vždy čtveřice souborů, z nichž první obsahuje určité množství vět jako takových, druhý jejich morfologickou anotaci, třetí analytickou a čtvrtý tektogramatickou. Proto také dále

[1] Přesněji PDT existuje i v dalších formátech, ale PML je z nich nejnovější a preferovaný.

v textu hovoříme o čtyřech rovinách PDT (první nazýváme *slovní* rovinou), v takovou chvíli se odkazujeme právě na tuto fyzickou reprezentaci. Každá věta (stejně jako každé slovo) má svůj identifikátor, pomocí kterého je možné v souborech hledat její metadata. Takovýchto čtveřic souborů obsahuje PDT několik tisíc, neboť byly-li by všechny věty umístěny do čtveřice jediné, nesmírně obtížně by se s ní manipulovalo, navíc veškerá práce by byla velmi náročná na systémové prostředky. Pro účely zpracování statistickými metodami jsou pak tyto čtveřice rozděleny do osmi trénovacích a dvou testovacích skupin.

Úplnou dokumentaci formátu PML lze nalézt v [7], my jej v následujících odstavcích nastíníme na příkladu věty s identifikátorem cmpr9415-025-p20s5: *Potom právo reklamace zaniká.*

4.2.1 Slovní rovina

Na slovní rovině je věta reprezentována takto:

```
<w id="w-cmpr9415-025-p20s5w1">
 <token>Potom</token>
</w>
<w id="w-cmpr9415-025-p20s5w2">
 <token>právo</token>
</w>
<w id="w-cmpr9415-025-p20s5w3">
 <token>reklamace</token>
</w>
<w id="w-cmpr9415-025-p20s5w4">
 <token>zaniká</token>
 <no_space_after>1</no_space_after>
</w>
<w id="w-cmpr9415-025-p20s5w5">
 <token>.</token>
</w>
```

Nás zajímají především atributy *id* elementů *w*, z nichž zjistíme identifikátor věty, a dále elementy *token* obsahující samotná slova věty.

4.2.2 Morfologická rovina

Na morfologické rovině je věta reprezentována takto:

```
<s id="m-cmpr9415-025-p20s5">
 <m id="m-cmpr9415-025-p20s5w1">
  <src.rf>manual</src.rf>
  <w.rf>w#w-cmpr9415-025-p20s5w1</w.rf>
  <form>Potom</form>
  <lemma>potom</lemma>
  <tag>Db-------------</tag>
 </m>
 <m id="m-cmpr9415-025-p20s5w2">
  <src.rf>manual</src.rf>
  <w.rf>w#w-cmpr9415-025-p20s5w2</w.rf>
  <form>právo</form>
  <lemma>právo_^(právo_na_něco;_také_jako_obor)</lemma>
  <tag>NNNS1-----A----</tag>
 </m>

 ...

</s>
```

Z atributu *id* elementu *s* zjistíme identifikátor věty, elementy *w.rf* obsahují reference na slova v souboru slovní roviny. V elementu *form* najdeme ve větě použitý slovní tvar, v elementu *lemma* tvar základní. Element *tag* obsahuje kompletní morfologické informace o slovu, podrobnosti viz [3].

4.2.3 Analytická rovina

Na analytické rovině je věta reprezentována takto:

```
<LM id="a-cmpr9415-025-p20s5">
 <s.rf>m#m-cmpr9415-025-p20s5</s.rf>
 <ord>0</ord>
 <children>
```

```
  <LM id="a-cmpr9415-025-p20s5w4">
   <m.rf>m#m-cmpr9415-025-p20s5w4</m.rf>
   <afun>Pred</afun>
   <ord>4</ord>
   <children>
    <LM id="a-cmpr9415-025-p20s5w1">
     <m.rf>m#m-cmpr9415-025-p20s5w1</m.rf>
     <afun>Adv</afun>
     <ord>1</ord>
    </LM>
    <LM id="a-cmpr9415-025-p20s5w2">
     <m.rf>m#m-cmpr9415-025-p20s5w2</m.rf>
     <afun>Sb</afun>
     <ord>2</ord>
     <children>
      <LM id="a-cmpr9415-025-p20s5w3">
       <m.rf>m#m-cmpr9415-025-p20s5w3</m.rf>
       <afun>Atr</afun>
       <ord>3</ord>
      </LM>
     </children>
    </LM>
   </children>
  </LM>
  <LM id="a-cmpr9415-025-p20s5w5">
   <m.rf>m#m-cmpr9415-025-p20s5w5</m.rf>
   <afun>AuxK</afun>
   <ord>5</ord>
  </LM>
 </children>
</LM>
```

V atributech *id* a elementech *m.rf* opět nalezneme identifikátory odpovídajících prvků na jiné rovině. Velmi důležitý je element *afun* obsahující analytickou funkci slova. Závislosti jednotlivých slov jsou vyjádřeny přímo XML strukturou, rekurzivním vnořováním elementů *LM*.

4.2.4 Tektogramatická rovina

Na tektogramatické rovině je věta reprezentována takto:

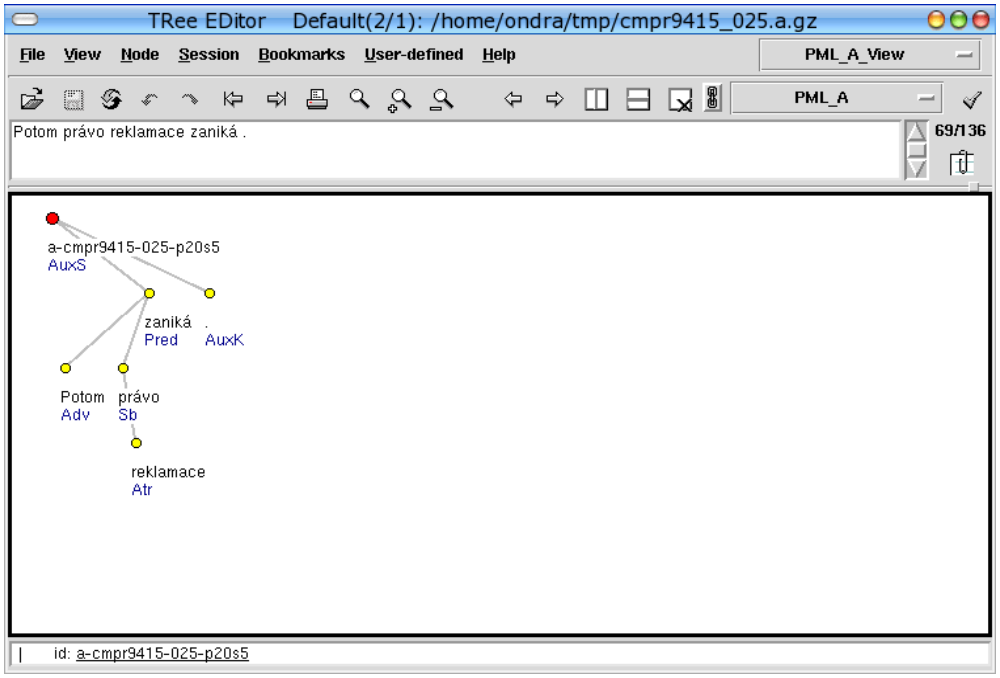
```
<LM id="t-cmpr9415-025-p20s5">
 <atree.rf>a#a-cmpr9415-025-p20s5</atree.rf>
 <nodetype>root</nodetype>
 <deepord>0</deepord>
 <children>
  <LM id="t-cmpr9415-025-p20s5w4">
   <a><lex.rf>a#a-cmpr9415-025-p20s5w4</lex.rf></a>
   <nodetype>complex</nodetype>
   <t_lemma>zanikat</t_lemma>
   <functor>PRED</functor>
   <tfa>f</tfa>
   <deepord>7</deepord>
   <sentmod>enunc</sentmod>
   <gram>
    <sempos>v</sempos>
    <verbmod>ind</verbmod>
    <deontmod>decl</deontmod>
    <tense>sim</tense>
    <aspect>proc</aspect>
    <resultative>res0</resultative>
    <dispmod>disp0</dispmod>
    <iterativeness>it0</iterativeness>
   </gram>
   <val_frame.rf>v#v-w8991f2</val_frame.rf>
   <children>
    <LM id="t-cmpr9415-025-p20s5w1">

    ...

    </LM>
   </children>
  </LM>
 </children>
</LM>
```

Kostrou své struktury je reprezentace tektogramatické roviny obdobná analytické, pouze obsahuje o jednotlivých uzlech výrazně větší množství

informací. Pro nás je nejdůležitější funktor obsažený v elementu *functor*, podrobný popis ostatních metadat viz [5].

4.3 Tree Editor TrEd

V souvislosti s daty PDT nelze nezmínit program TrEd ([6]). Jedná se o aplikaci určenou k prohlížení a editaci stromových struktur, jako jsou například právě závislostní stromy. Právě TrEd byl používán pro anotování PDT. My jsme při práci na cvičebnici využívali pouze zlomku jeho možností – vyhledávání v datech a zobrazování analytických a tektogramatických stromů, přesto bez něj by naše práce byla mnohem těžší.



Obrázek 4.1: Tree Editor TrEd

Kapitola 5

Filtrování vět

Pražský závislostní korpus obsahuje celou řadu vět nevhodných k procvičování žáků – vět obsahujících takové jevy, na jejichž klasifikacích se různé školské učebnice neshodují či je dokonce nezmiňují vůbec (ať už pro jejich komplexnost, nebo pro jejich okrajovost). Takové věty tedy bylo nutno z datového vzorku odstranit (protože není možné studenty procvičovat v látce, o níž se neučili), pochopitelně automatickou cestou.

5.1 Program FilterSentences

Program *FilterSentences* slouží k filtrování vět PDT. S korpusem pracuje v jeho nejnovějším formátu, PML (viz kapitolu 4), a to jak na vstupu, tak i na výstupu. Pro každou vstupní větu načte informace o jejím označkování na slovní, morfologické, analytické a tektogramatické rovině a tyto předá zvolenému *filtračnímu kritériu*. Filtrační kritérium rozhodne, zda mu daná věta odpovídá, či ne, a na základě tohoto výsledku program větu zařadí, nebo nezařadí do výstupních dat.

5.2 Existující filtrační kritéria

Následující odstavce popisují filtrační kritéria, jež byla na vstupní data použita. U každého z nich je uvedeno, na základě čeho (a jak) k filtrování dochází, a je přiložen příklad věty, která kritérium nesplňuje. Rovněž

je zobrazena grafická reprezentace analytické roviny věty získaná z programu TrEd ([6]). Jednotlivé filtry jsou uvedeny v tom pořadí, v jakém byly postupně aplikovány, kdy výstupní množina dat jednoho filtru se stala vstupní množinou dat filtru následujícího. Postupné ubývání vět v datovém vzorku dokládají tabulky uzavírající popis filtrů. Data PDT jsou pro účely statistických metod zpracování a ověřování jejich účinnosti rozděleny na deset částí (dtest, etest, train-1, . . . , train-8), my jsme toto rozdělení při práci s nimi zachovali, proto tabulky obsahují kromě čísel odpovídajících celé množině dat i údaje z těchto jednotlivých částí.

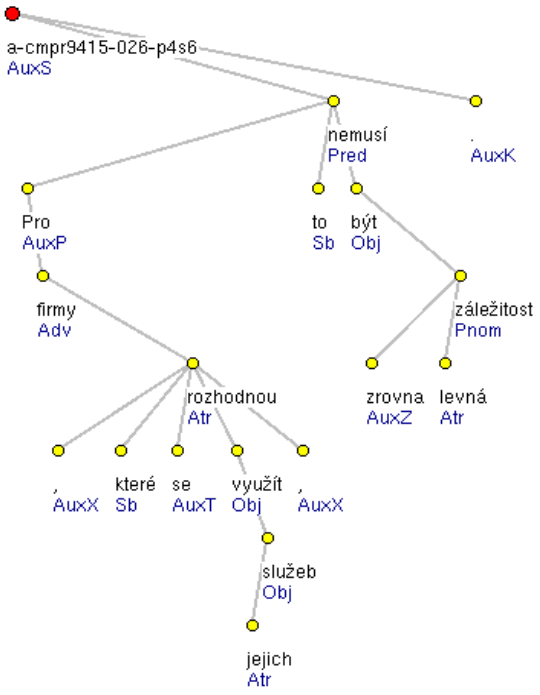
5.2.1 SimpleSentence

Toto kritérium odstraňuje všechna souvětí – souřadná i podřadná – a ponechává tak pouze jednoduché věty.

Testování, zda je věta souvětím, je vůbec nejsložitější filtr kvůli množství případů, které musí postihnout. Probíhá trojím způsobem. Nejprve se na analytické rovině zjišťuje, zda věta obsahuje koordinaci, jejímž členem je predikát, pokud ano, je věta prohlášena za souvětí. Tímto způsobem jsou odhalena běžná souřadná souvětí. Dále se ve větě hledají všechny čárky. Pokud existuje taková čárka, která má analytickou funkci *AuxX*, přitom však její rodič nemá funkci *Coord*, je věta prohlášena za souvětí. Tato situace nastává pro běžné typy podřadných souvětí (například jedna hlavní věta a jedna na ní závislá vedlejší věta). Nakonec jsou znovu prohledány všechny koordinace ve větě, tentokrát ale na tektogramatické rovině. Pokud existuje taková, že její funktor je jeden z *CONJ*, *ADVS*, *CSQ*, *DISJ*, *GRAD*, *REAS*, *CONFR*, *CONTRA*, *OPER*, *APPS* a že alespoň jeden její člen má v gramatému *tense* jednu z hodnot *post*, *ant*, *sim*, *nir*, je věta prohlášena za souvětí. Toto pravidlo postihuje zbylé typy souvětí, především situaci, kdy je hlavní věta rozvinutá dvěma vedlejšími koordinovanými větami.

Příklad odstraněné věty:
Pro firmy, které se rozhodnou využít jejich služeb, to nemusí být zrovna levná záležitost. (cmpr9415-026-p4s6[1])

[1] Identifikátor věty v datech PDT.



Obrázek 5.1: Věta vyřazená na základě filtračního kritéria *SimpleSentence*

Sada	Celkem	Zachováno	Vyřazeno	Zachováno
dtest	5228	2384	2844	45,6 %
etest	5476	2419	3057	44,2 %
train-1	4709	2204	2505	46,8 %
train-2	4790	2301	2489	48,0 %
train-3	5064	2272	2792	44,9 %
train-4	4418	1953	2465	44,2 %
train-5	5027	2320	2707	46,2 %
train-6	4870	2162	2708	44,4 %
train-7	4724	2214	2510	46,9 %
train-8	5136	2323	2813	45,2 %
Celkem	**49442**	**22552**	**26890**	**45,6 %**

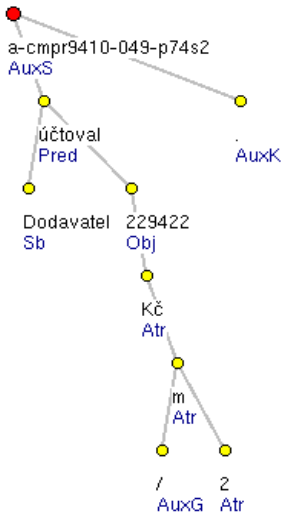
Tabulka 5.1: Statistika vět zachovaných a vyřazených filtračním kritériem *SimpleSentence*

5.2.2 GraphicalSymbols

Toto kritérium odstraňuje všechny věty, které obsahují nejrůznější grafické symboly (analytická funkce *AuxG* – např. dvojtečky, uvozovky, závorky, hvězdičky a další), neboť drtivá většina takovýchto vět obsahuje konstrukce školskými prostředky nepopsatelné. Výjimku tvoří případy, kdy analytickou funkci *AuxG* nese tečka.

Filtr si vystačí s informacemi z morfologické a analytické roviny – ke všem uzlům s analytickou funkcí *AuxG* hledá slovní tvar a pokud jím není tečka, věta je vyřazena.

Příklad odstraněné věty:
Dodavatel účtoval 229422 Kč/m^2. (cmpr9410-049-p74s2)



Obrázek 5.2: Věta vyřazená na základě filtračního kritéria *GraphicalSymbols*

5.2.3 EllipsisAposition

Toto kritérium odstraňuje věty obsahující elipsy (*ExD*) nebo apozice (*Apos*). Pracuje pouze s informacemi z analytické roviny, kdykoliv nalezne uzel s analytickou funkcí *ExD* či *Apos*, je věta vyřazena.

Sada	Celkem	Zachováno	Vyřazeno	Zachováno
dtest	2384	2151	233	90,2 %
etest	2419	2180	239	90,1 %
train-1	2204	1987	217	90,2 %
train-2	2301	2070	231	90,0 %
train-3	2272	2066	206	90,9 %
train-4	1953	1769	184	90,6 %
train-5	2320	2088	232	90,0 %
train-6	2162	1948	214	90,1 %
train-7	2214	2011	203	90,8 %
train-8	2323	2114	209	91,0 %
Celkem	**22552**	20384	2168	**90,4 %**

Tabulka 5.2: Statistika vět zachovaných a vyřazených filtračním kritériem *GraphicalSymbols*

Příklad odstraněné věty:
8) Stejné požadavky jako na dovážené výrobky bude SPPI uplatňovat i u výrobků tuzemských. (cmpr9410-001-p22s1)

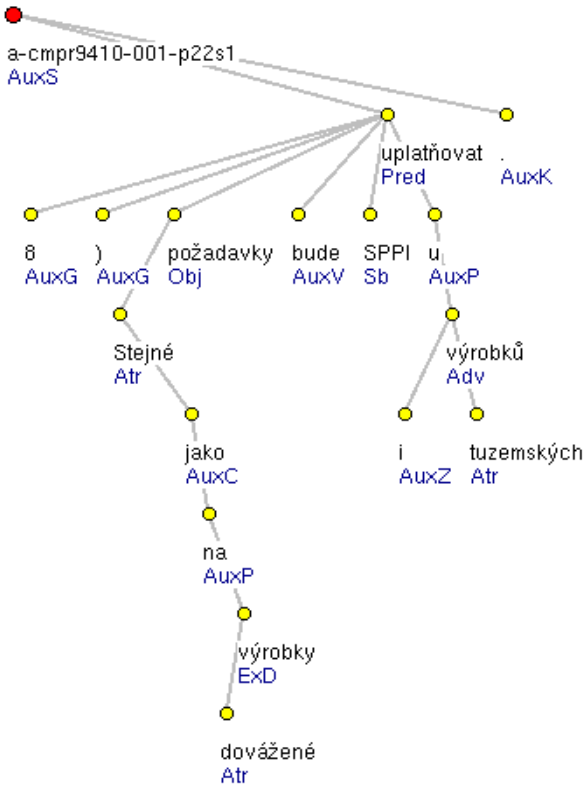
5.2.4 OnePredicate

Toto kritérium vybírá věty, které obsahují právě jeden přísudek (*Pred*), tedy především odstraní věty zcela bez přísudku (věty s více než jedním přísudkem již odstranilo kritérium *SimpleSentence*). I tomuto filtru stačí informace z analytické roviny, prochází se zde všechny analytické funkce a počítají se predikáty.

Příklad odstraněné věty:
Nová striktní omezení vlády SR proti českým exportérům (cmpr9410-001-p3s1)

5.2.5 LessThanNWords

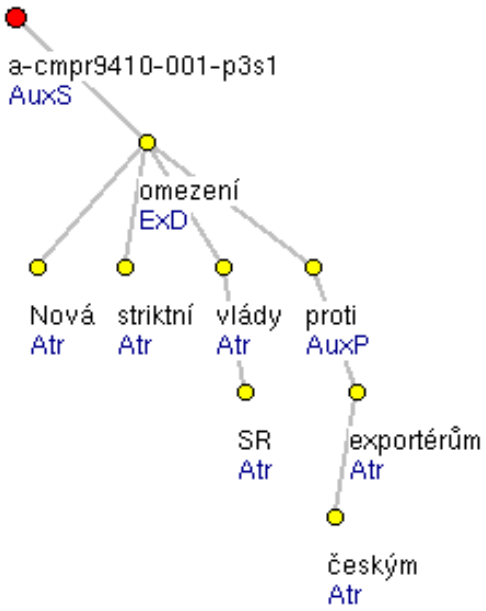
Toto kritérium vybírá věty, v nichž celkový počet slov nepřesáhne množství určené konstantou *MAX_COUNT* (její hodnotu jsme se rozhodli



Obrázek 5.3: Věta vyřazená na základě filtračního kritéria *EllipsisAposition*

Sada	Celkem	Zachováno	Vyřazeno	Zachováno
dtest	2151	1506	645	70,0 %
etest	2180	1444	736	66,2 %
train-1	1987	1311	676	66,0 %
train-2	2070	1418	652	68,5 %
train-3	2066	1413	653	68,4 %
train-4	1769	1162	607	65,7 %
train-5	2088	1400	688	67,0 %
train-6	1948	1239	709	63,6 %
train-7	2011	1315	696	65,4 %
train-8	2114	1425	689	67,4 %
Celkem	20384	13633	6751	**66,9 %**

Tabulka 5.3: Statistika vět zachovaných a vyřazených filtračním kritériem *EllipsisAposition*



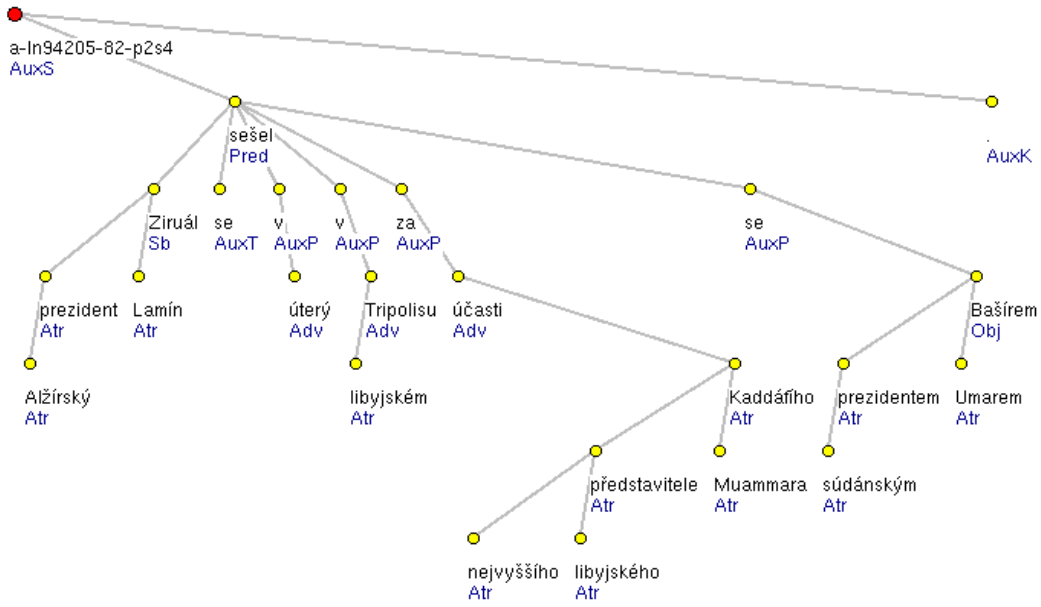
Obrázek 5.4: Věta vyřazená na základě filtračního kritéria *OnePredicate*

Sada	Celkem	Zachováno	Vyřazeno	Zachováno
dtest	1506	1505	1	99,9 %
etest	1444	1439	5	99,7 %
train-1	1311	1308	3	99,8 %
train-2	1418	1416	2	99,9 %
train-3	1413	1413	0	100,0 %
train-4	1162	1160	2	99,8 %
train-5	1400	1400	0	100,0 %
train-6	1239	1237	2	99,8 %
train-7	1315	1315	0	100,0 %
train-8	1425	1424	1	99,9 %
Celkem	13633	13617	16	**99,9 %**

Tabulka 5.4: Statistiky vět zachovaných a vyřazených filtračním kritériem *OnePredicate*

nastavit na 19). Důvodem pro použití tohoto filtru je fakt, že příliš dlouhé věty mohou svojí stavbou a složitostí žáky zbytečně mást, navíc největší část takovýchto vět tvoří dlouhé řetězce shodných či neshodných přívlastků, které jsou z hlediska procvičování „nezajímavé“. Filtr je aplikován na analytické rovině, ale stejně dobře by ho šlo použít na rovině morfologické nebo slovní.

Příklad odstraněné věty:
Alžírský prezident Lamín Ziruál se sešel v úterý v libyjském Tripolisu za účasti nejvyššího libyjského představitele Muammara Kaddáfího se súdánským prezidentem Umarem Bašírem. (ln94205-82-p2s4)



Obrázek 5.5: Věta vyřazená na základě filtračního kritéria *LessThanNWords*

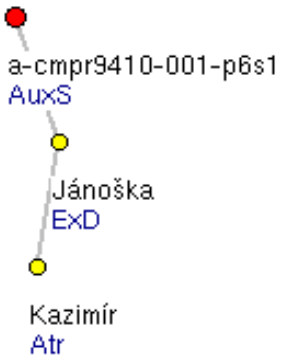
5.2.6 MoreThanNWords

Toto kritérium vybírá věty, v nichž celkový počet slov je roven alespoň *MIN_COUNT* (tuto hodnotu jsme se rozhodli nastavit na 5). Tím dojde k vyloučení naopak příliš jednoduchých vět, mnohdy jenom krátkých nadpisů a podobně. Filtr je aplikován na analytické rovině, ale stejně dobře by ho šlo použít na rovině morfologické nebo slovní.

Sada	Celkem	Zachováno	Vyřazeno	Zachováno
dtest	1505	1442	63	95,8 %
etest	1439	1382	57	96,0 %
train-1	1308	1245	63	95,2 %
train-2	1416	1352	64	95,5 %
train-3	1413	1353	60	95,8 %
train-4	1160	1102	58	95,0 %
train-5	1400	1339	61	95,6 %
train-6	1237	1184	53	95,7 %
train-7	1315	1252	63	95,2 %
train-8	1424	1359	65	95,4 %
Celkem	13617	13010	607	**95,5 %**

Tabulka 5.5: Statistiky vět zachovaných a vyřazených filtračním kritériem *LessThanNWords*

Příklad odstraněné věty:
Kazimír Jánoška (cmpr9410-001-p6s1)



Obrázek 5.6: Věta vyřazená na základě filtračního kritéria *MoreThanNWords*

5.2.7 AuxO

Toto kritérium odstraňuje věty, které obsahují slovo s analytickou funkcí *AuxO* – nadbytečný (odkazovací, emotivní) element. Jedinou výjimku tvo-

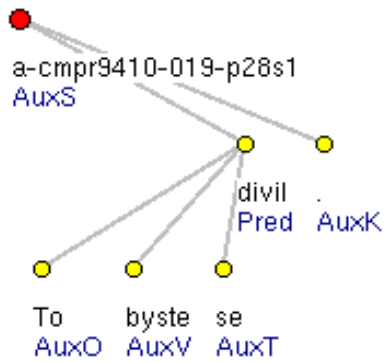
Sada	Celkem	Zachováno	Vyřazeno	Zachováno
dtest	1442	1277	165	88,6 %
etest	1382	1229	153	88,9 %
train-1	1245	1121	124	90,0 %
train-2	1352	1217	135	90,0 %
train-3	1353	1217	136	89,9 %
train-4	1102	1005	97	91,2 %
train-5	1339	1228	111	91,7 %
train-6	1184	1071	113	90,5 %
train-7	1252	1131	121	90,3 %
train-8	1359	1222	137	89,9 %
Celkem	13010	11718	1292	**90,1 %**

Tabulka 5.6: Statistiky vět zachovaných a vyřazených filtračním kritériem *MoreThanNWords*

ří částice *si*, u té nám funkce *AuxO* nevadí.

Technicky je filtr proveden analogicky k filtru *GraphicalSymbols* – hledá slovní tvary k uzlům s analytickou funkcí *AuxO* a pokud aspoň jeden z nich není *si*, je věta vyřazena.

Příklad odstraněné věty:
To byste se divil. (cmpr9410-019-p28s1)



Obrázek 5.7: Věta vyřazená na základě filtračního kritéria *AuxO*

Sada	Celkem	Zachováno	Vyřazeno	Zachováno
dtest	1277	1276	1	99,9 %
etest	1229	1228	1	99,9 %
train-1	1121	1120	1	99,9 %
train-2	1217	1215	2	99,8 %
train-3	1217	1216	1	99,9 %
train-4	1005	1005	0	100,0 %
train-5	1228	1226	2	99,8 %
train-6	1071	1067	4	99,6 %
train-7	1131	1130	1	99,9 %
train-8	1222	1222	0	100,0 %
Celkem	11718	11705	13	**99,9 %**

Tabulka 5.7: Statistiky vět zachovaných a vyřazených filtračním kritériem *AuxO*

5.2.8 IndividualSentences

Toto kritérium není postaveno na žádném lingvistickém základě. Věty vyřazuje podle identifikátorů nalezených v konfiguračním souboru. Existuje proto, abychom mohli v případě potřeby odstranit některé věty, které nedokážeme dobře ve cvičebnici zobrazit, ale pro které se nám zatím nepodařilo nalézt nějakou obecnější charakteristiku, na jejímž základě bychom mohli vytvořit plnohodnotné filtrační kritérium.

5.2.9 KeepAll

Toto kritérium existuje především z technických důvodů, vybírá jakoukoliv předloženou větu, žádnou neodmítne.

5.3 Použitá filtrační kritéria

Jak je vidět i z předchozích příkladů, často nastává situace, že se jednotlivé filtry překrývají a mnohé věty by tak byly vyřazeny na základě hned několika z nich, avšak ani jedno z uvedených kritérií není pouze speciálním případem jiného.

Výchozím a nejdůležitějším bylo kritérium *SimpleSentence*, neboť ani na základních, ani na středních školách se neučí, jak souvětí na analytické rovině zpracovávat. Zvažovali jsme samozřejmě, zda (alespoň některá) souvětí nezachovat pro procvičování morfologie. Ukázalo se však, že to není potřeba, neboť toto kritérium zachovalo přibližně 46 % všech vět. Datový vzorek se tedy nezredukoval na takové množství, které by nám nepřipadalo dostatečně bohaté.

Z původních 49 442 vět, které do filtračního procesu vstupovaly, jich všemi filtry prošlo 11 705, tedy asi 24 %. Je to číslo o něco menší, než v jaké jsme na začátku doufali, ale stále ještě poskytuje velice široký výběr vět pro cvičení.

Kapitola 6

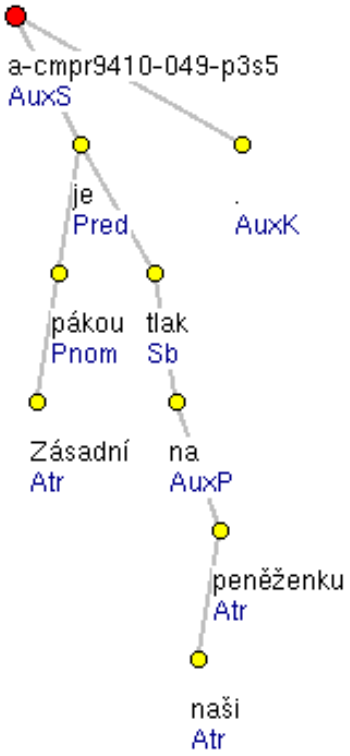
Transformace syntaktických stromů

Dalším krokem po filtraci pro naše účely nepoužitelných vět byla transformace informací získaných z PDT do podoby, se kterou by uživatelé byli schopni snadno pracovat. Zatímco na úrovni morfologické toto nevyžadovalo většího úsilí, bylo již předem jasné, že na úrovni syntaktické bude nutné množství úprav.

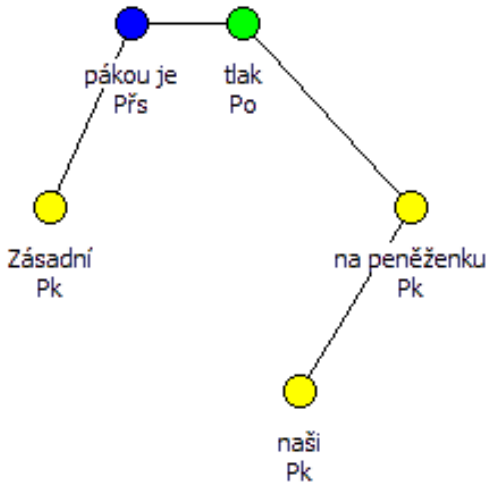
Analytická rovina PDT se totiž v nemálo ohledech značně liší od syntaxe tak, jak se učí ve školách. Předně obsahuje mnoho analytických funkcí, ke kterým neexistují odpovídající větné členy (a naopak některé informace větných členů se týkající jsou v PDT obsaženy až na rovině tektogramatické). Za druhé na ní každému slovu věty (včetně interpunkce) odpovídá právě jeden uzel, zatímco ve školské reprezentaci může obsahovat (a velice často také obsahuje) jediný uzel slov několik. Celému uzlu (nebo chceme-li všem těmto slovům) pak přísluší pouze jeden větný člen.

Situaci demonstrují následující dva obrázky, na kterých je zachycena věta *„Zásadní pákou je tlak na naši peněženku.“* (cmpr9410-049-p3s5) – nejprve se jedná o zobrazení analytické roviny programem TrEd, potom o zobrazení programem Charon po provedení potřebných transformací. V dalším textu budeme o těchto dvou pohledech na věty hovořit jako o *akademickém rozboru*, respektive *školském rozboru*.

Naše původní představa byla ta, že úprav analytické roviny nebude potřeba mnoho a že postačí několik jednoduchých transformačních procedur,



Obrázek 6.1: Akademický rozbor věty *„Zásadní pákou je tlak na naši peněženku.“*



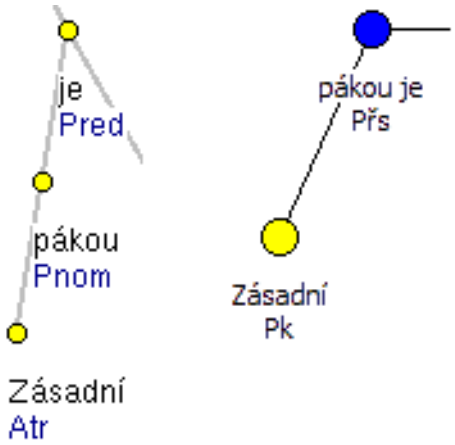
Obrázek 6.2: Školský rozbor věty *„Zásadní pákou je tlak na naši peněženku.“*

jež vyplynou přímo z nahlédnutí do dat PDT. Brzy se však tato myšlenka ukázala mylnou a byli jsme nuceni použít jiného, pečlivějšího přístupu. Tím bylo prostudování celého analytického manuálu PDT ([2]), rozebrání všech situací, o kterých se zmiňuje, posouzení, jak se jejich popis liší od výuky ve školách, a vyvození závěrů, jak případné rozdíly řešit.

V následující části textu budeme tyto situace procházet a budeme se tedy velice často na analytický manuál, který je stěžejním zdrojem informací pro tuto práci, odkazovat.

Výchozí syntaktickou reprezentací věty je strom tak, jak se nachází v datech PDT, tedy takový, kde každému slovu odpovídá jeden uzel a jedna analytická funkce a kde každý uzel (s výjimkou technického kořene) má právě jednoho rodiče. Výsledná reprezentace je obecnější v tom smyslu, že uzel může obsahovat více než jenom jedno slovo. Nad uzly stromu zavádíme následující operace, které při transformacích používáme:

- Připojení k rodiči. Slova obsažená v uzlu jsou přesunuta do rodičovského uzlu, děti uzlu se stávají dětmi rodiče uzlu, uzel samotný zaniká (viz obrázek 6.3).

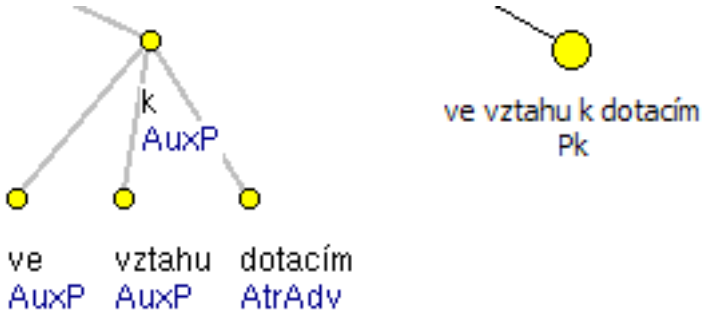


Obrázek 6.3: Operace *připojení k rodiči*

V levé části obrázku je situace před provedením operace *připojení k rodiči* na uzlu *pákou*, v pravé části je zobrazen výsledek po transformaci.

- Pohlcení dětí. Všechna slova všech dětí jsou přesunuta do tohoto uzlu, všechny děti dětí tohoto uzlu se stávají novými dětmi uzlu,

původní děti uzlu zanikají (viz obrázek 6.4). Operace je ekvivalentní provedení operace *připojení k rodiči* na všech dětech uzlu.



Obrázek 6.4: Operace *pohlcení dětí*

V levé části obrázku je situace před provedením operace *pohlcení dětí* na uzlu *k*, v pravé části je zobrazen výsledek po transformaci.

- Odstranění uzlu. Uzel je ze stromu odstraněn. Tuto operaci je možné použít pouze na listy stromu.

6.1 Analytická funkce *Pred*

Touto funkcí je označen „predikát, resp. uzel, který nezávisí na jiném uzlu; věší se na #“. Podrobný popis viz [2], oddíl 3.3.1.

V námi tvořené školské reprezentaci tomuto uzlu přisoudíme větný člen *přísudek slovesný* a dále se jím nezabýváme (ponecháme jej zavěšen pod technickým kořenem celé věty). Je-li potřeba větný člen změnit (v případě, že se ve skutečnosti jedná o přísudek jmenný), stane se tak při zpracovávání některého z ostatních uzlů.

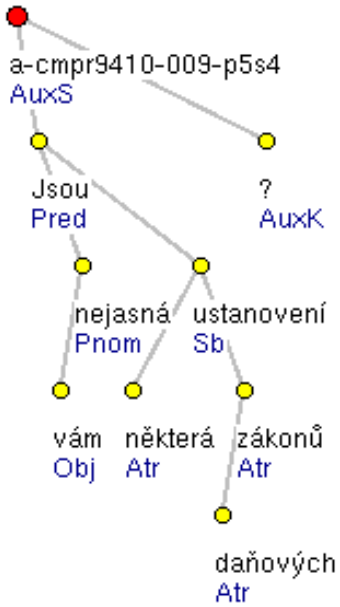
6.2 Analytická funkce *Pnom*

Touto funkcí je označen „predikát nominální, resp. jmenná část přísudku se sponou být“. Podrobný popis viz [2], oddíl 3.3.1.

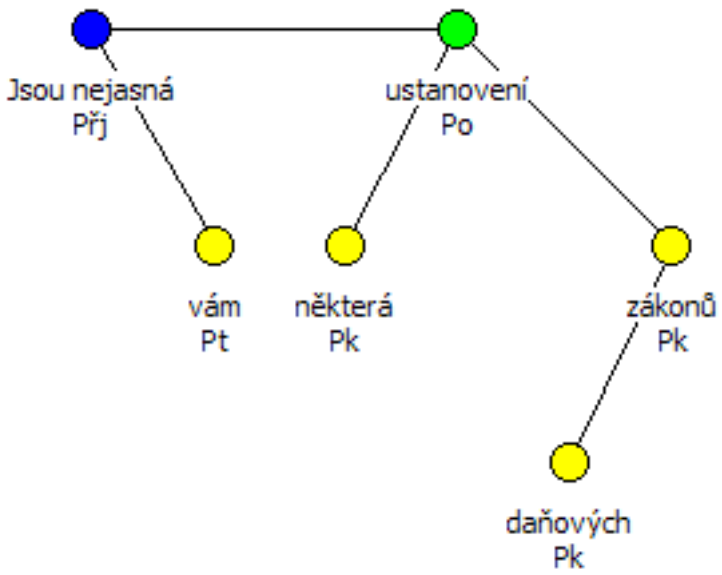
Tento uzel je ve školské reprezentaci chápán jako součást přísudku, je tedy nutno použít následující transformaci: dokud rodičem uzlu není uzel s funkcí *Coord* nebo *Pred*, provádět operaci *připojení k rodiči*.

- Potom je-li rodičem uzlu uzel s funkcí *Coord*, ještě jednou provést operaci *připojení k rodiči* a u rodiče si poznamenat, že koordinuje jmennou část přísudku. Více o zpracování koordinace v části 6.22.
- Jinak (to znamená, že rodičovský uzel má funkci *Pred*) ještě jednou provést operaci *připojení k rodiči* a změnit rodiči větný člen na *přísudek jmenný*.

Příklad:
Jsou (Pred) *vám nejasná* (Pnom) *některá ustanovení daňových zákonů?*
(cmpr9410-009-p5s4)
Uzel *nejasná* je připojen k uzlu *Jsou* a společně tvoří *přísudek jmenný*. (Viz obrázky 6.5 a 6.6.)



Obrázek 6.5: Zachycení analytické funkce *Pnom* v akademickém rozboru



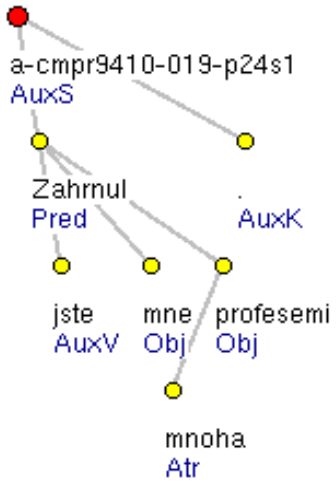
Obrázek 6.6: Zachycení jmenného přísudku ve školském rozboru

6.3 Analytická funkce *AuxV*

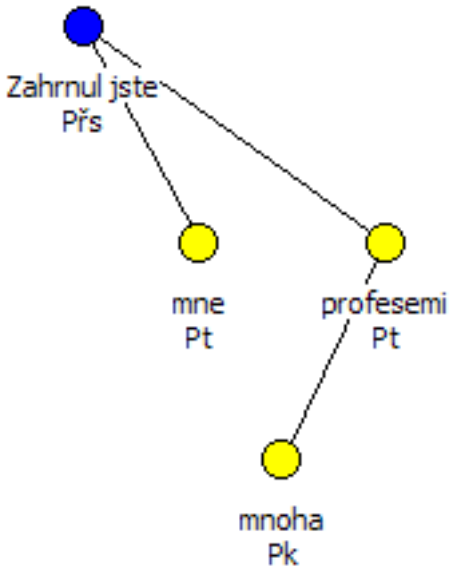
Touto funkcí je označeno „pomocné sloveso být (Auxiliary Verb)“. Podrobný popis viz [2], oddíl 3.3.1.

Tento uzel je rovněž chápán jako součást přísudku, je tedy nutno jej transformovat pomocí operace *připojení k rodiči*. Charakter samotného přísudku se tím nemění, o tom, zda je přísudek slovesný, či jmenný, tento uzel nerozhoduje. Pouze je nutné pamatovat na to, že tento uzel nemusí být dítětem pouze uzlu s funkcí *Pred*, ale jeho rodičem může být i jiný uzel s funkcí *AuxV*, a že v takovém případě je potřeba touž transformaci provést i s tímto rodičem.

Příklad:
Zahrnul (Pred) *jste* (AuxV) *mne mnoha profesemi.* (cmpr9410-019-p24s1)
Uzel *jste* je připojen k uzlu *Zahrnul* a společně tvoří *přísudek slovesný*. (Viz obrázky 6.7 a 6.8.)



Obrázek 6.7: Zachycení analytické funkce *AuxV* v akademickém rozboru



Obrázek 6.8: Zachycení složeného slovesného přísudku ve školském rozboru

6.4 Analytická funkce *Sb*

Touto funkcí je označen „subjekt (podmět)“. Podrobný popis viz [2], oddíl 3.3.2.

Ve školách tvoří výchozí bod výuky syntaxe pojem *základní skladební dvojice*, tu tvoří podmět s přísudkem. Tyto dva větné členy jsou postaveny na stejnou úroveň, a tak školské syntaktické stromy nejsou stromy v pravém slova smyslu, protože nemají kořen (alternativně se na to můžeme dívat tak, že mají kořeny dva). Proto je potřeba provést následující transformaci:

- Je-li rodičem uzlu uzel s funkcí *Coord*, provést operaci *připojení k rodiči* a u rodiče si poznamenat, že koordinuje podmět. Více o zpracování koordinace v části 6.22.
- Jinak (to znamená, že rodičovský uzel má funkci *Pred*) uzel odtrhnout od rodiče, zavěsit ho přímo pod technický kořen celé věty a přisoudit mu větný člen *podmět*.

Příklad:
Lidé (Sb) *prožívají* (Pred) *nebývale nervózní dobu*. (cmpr9410-019-p34s1)
Uzel *Lidé* je povýšen na úroveň uzlu *prožívají*. (Viz obrázky 6.9 a 6.10.)

6.5 Analytická funkce *Atr*

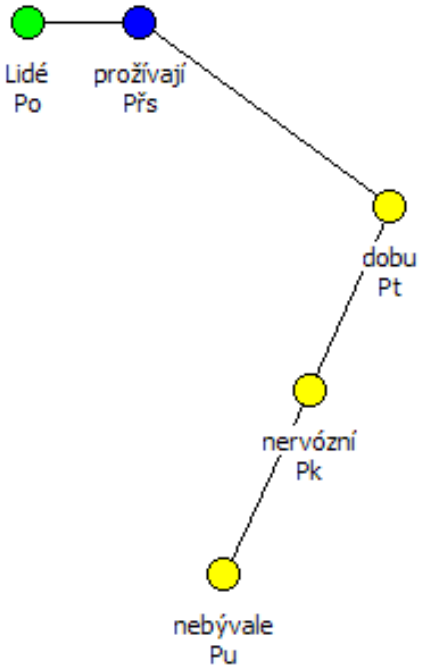
Touto funkcí je označen „atribut (přívlastek)“. Podrobný popis viz [2], oddíl 3.3.3.

Funkce *Atr* přímo odpovídá větnému členu *přívlastek*, není tedy takovýto uzel potřeba nijak transformovat.

6.6 Analytické funkce *AtrAdv*, *AdvAtr*, *AtrAtr*, *AtrObj*, *ObjAtr*

Všechny tyto funkce jsou speciálními typy přívlastků a je proto s nimi nakládáno, jako by se jednalo o funkci *Atr.* Viz oddíl 6.5.

Obrázek 6.9: Zachycení analytické funkce *Sb* v akademickém rozboru



Obrázek 6.10: Zachycení podmětu ve školském rozboru

6.7 Analytická funkce *Obj*

Touto funkcí je označen „objekt (předmět)“. Podrobný popis viz [2], oddíl 3.3.4.

Označení uzlu v PDT za objekt se plně nekryje s chápáním předmětu ve školách, pokrývá i další dva případy, které je potřeba postihnout v následující transformaci:

- Je-li objekt z hlediska morfologie sloveso v infinitivu a je-li jeho rodič modální sloveso, provést operaci *připojení k rodiči* (uzel se tak stane součástí slovesného přísudku).
- Má-li objekt na tektogramatické rovině funktor *EFF* a z hlediska morfologie se jedná o podstatné jméno
  - v prvním pádě
  - v sedmém pádě
  - nebo ve čtvrtém pádě po předložce *za*,

  jedná se ve skutečnosti z hlediska školské výuky o *doplněk* a je potřeba uzlu tento větný člen přiřadit. Tato transformace není zatím v systému implementována (viz oddíl 8.2).
- Jinak pouze uzlu přiřadit větný člen *předmět*.

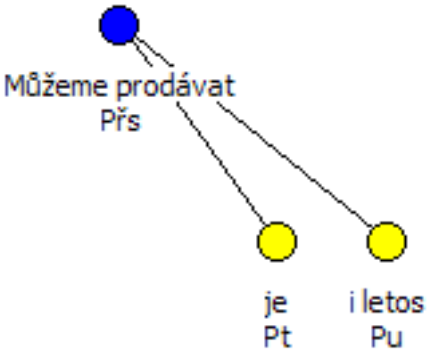
Příklad:
Můžeme (Pred) *je prodávat* (Obj) *i letos?* (cmpr9413-007-p3s2)
Uzel *prodávat* je připojen k uzlu *Můžeme* a společně tvoří *přísudek slovesný*. (Viz obrázky 6.11 a 6.12.)

6.8 Analytická funkce *Adv*

Touto funkcí je označeno „adverbiale (příslovečné určení, bez dalšího rozlišení)“. Podrobný popis viz [2], oddíl 3.3.5.

Funkce *Adv* přímo odpovídá větnému členu *příslovečné určení*, není tedy takovýto uzel potřeba nijak transformovat.

Obrázek 6.11: Zachycení analytické funkce *Obj* v akademickém rozboru



Obrázek 6.12: Zachycení slovesného přísudku tvořeného modálním slovesem a slovesem v infinitivu ve školském rozboru

6.9 Analytická funkce *Atv*

Touto funkcí je označen „doplněk (jen tzv. určující), technicky zavěšen na neslovesném členu“. Podrobný popis viz [2], oddíl 3.3.6.

Funkce *Atv* přímo odpovídá větnému členu *doplněk*, není tedy takovýto uzel potřeba nijak transformovat.

6.10 Analytická funkce *AtvV*

Touto funkcí je označen „doplněk (jen tzv. určující), visící na slovese (chybí druhý řídící člen)“. Podrobný popis viz [2], oddíl 3.3.6.

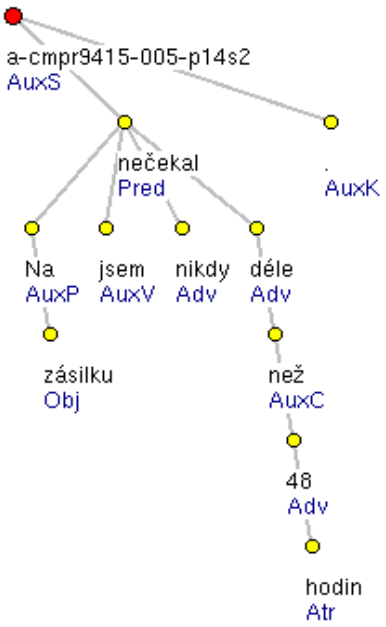
Funkce *AtvV* přímo odpovídá větnému členu *doplněk*, není tedy takovýto uzel potřeba nijak transformovat.

6.11 Analytická funkce *AuxC*

Touto funkcí je označena „spojka (podřadící)“. Podrobný popis viz [2], oddíl 3.3.7.1.

Nejčastěji uzel s funkcí *AuxC* uvozuje vedlejší větu, všechna souvětí jsme ovšem z dat vyřadili použitím filtru *SimpleSentence* (viz oddíl 5.2.1). Ve zbylých případech, kdy uzel uvozuje pouze větný člen, je na něm nutné provést operaci *pohlcení dětí*. Právě jedno z původních dětí mělo analytickou funkci přímo odpovídající některému větnému členu, tímto větným členem bude uzel označen.

Příklad:
Na zásilku jsem nečekal nikdy déle než (AuxC) *48* (Adv) *hodin.* (cmpr9415-005-p14s2)
Uzel *než* pohltí uzel *48* a společně tvoří *příslovečné určení*. (Viz obrázky 6.13 a 6.14.)



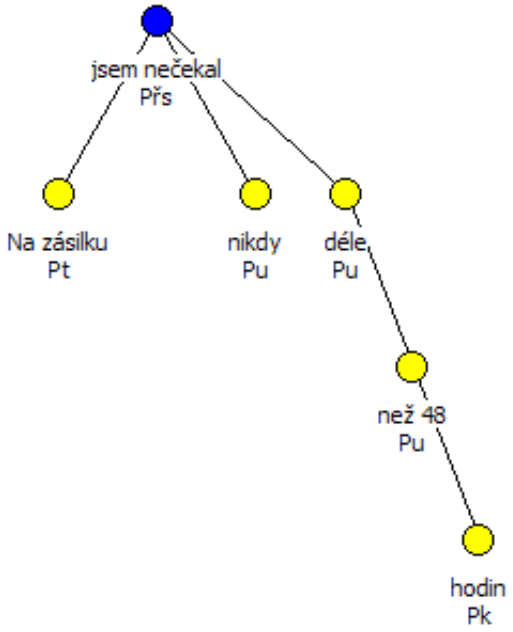
Obrázek 6.13: Zachycení analytické funkce *AuxC* v akademickém rozboru

6.12 Analytická funkce *AuxP*

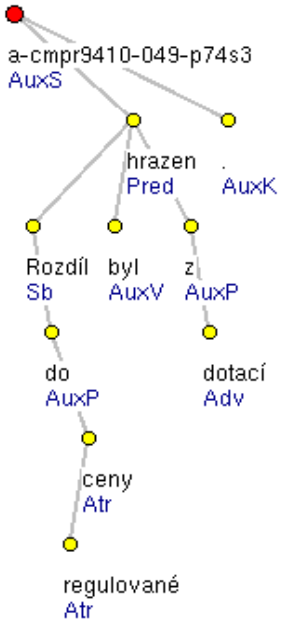
Touto funkcí je označena „předložka primární, části předložky sekundární“. Podrobný popis viz [2], oddíl 3.3.7.2.

Na uzlu s funkcí *AuxP* je nutné provést operaci *pohlcení dětí*. Pokud uzel vůbec nějaké děti měl (tj. nebyl sám dítětem uzlu s funkcí *AuxP*), právě jedno z nich mělo analytickou funkci přímo odpovídající některému větnému členu a tímto větným členem bude uzel označen.

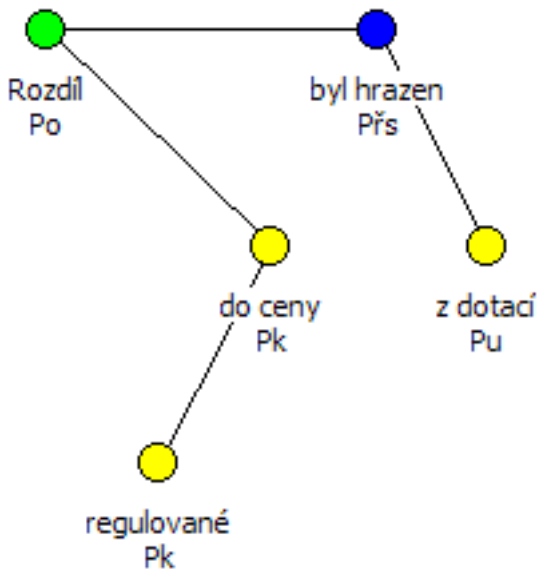
Příklad:
Rozdíl do (AuxP) *regulované ceny* (Atr) *byl hrazen z* (AuxP) *dotací* (Adv).
(cmpr9410-049-p74s3)
Uzel *do* pohltí uzel *ceny* a společně tvoří *přívlastek*, uzel *z* pohltí uzel *dotací* a společně tvoří *příslovečné určení*. (Viz obrázky 6.15 a 6.16.)



Obrázek 6.14: Výsledek transformace analytické funkce *AuxC* ve školském rozboru



Obrázek 6.15: Zachycení analytické funkce *AuxP* v akademickém rozboru



Obrázek 6.16: Výsledek transformace analytické funkce *AuxP* ve školském rozboru

6.13 Analytická funkce *AuxZ*

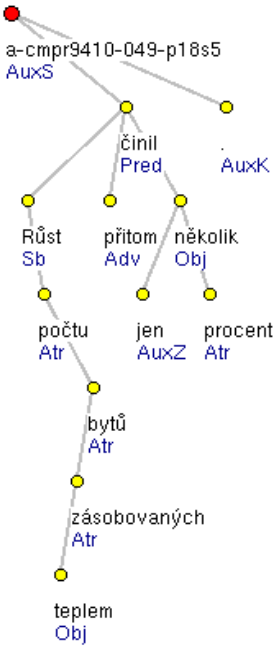
Touto funkcí je označeno „zdůrazňovací slovo“. Podrobný popis viz [2], oddíl 3.3.7.3.

Na uzlu s funkcí *AuxZ* je nutné provést operaci *připojení k rodiči*. Je-li rodičem uzel s funkcí *Coord*, musíme si u něj poznamenat, že koordinuje zdůrazňovací slovo (více o zpracování koordinace v části 6.22).

Příklad:
Růst počtu teplem zásobovaných bytů přitom činil jen (AuxZ) *několik* (Obj) *procent.* (cmpr9410-049-p18s5)
Uzel *jen* je připojen k uzlu *několik*. (Viz obrázky 6.17 a 6.18.)

6.14 Analytická funkce *AuxO*

Touto funkcí je označen „nadbytečný (odkazovací, emotivní) element“. Podrobný popis viz [2], oddíl 3.3.7.4.



Obrázek 6.17: Zachycení analytické funkce *AuxZ* v akademickém rozboru

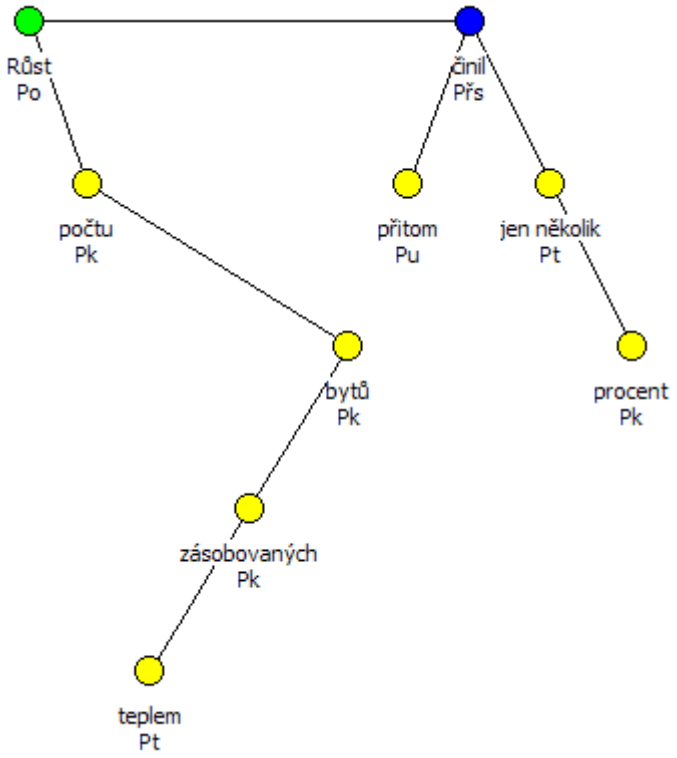
Jak jsme uvedli v části 5.2.7, věty obsahující uzly s funkcí *AuxO* z dat vyřazujeme, výjimku tvoří pouze částice *si*. Proto když při transformacích na funkci *AuxO* narazíme, víme, že se jedná o tuto situaci, a můžeme na uzlu provést operaci *připojení k rodiči*, protože pro naše účely se tento případ neliší od případů popsaných v částech 6.15 a 6.16.

Příklad:
Uvedeme (Pred) *si* (AuxO) *běžný příklad.* (ln94202-55-p6s1)
Uzel *si* je připojen k uzlu *Uvedeme*. (Viz obrázky 6.19 a 6.20.)

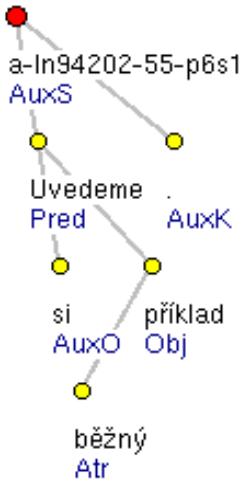
6.15 Analytická funkce *AuxT*

Touto funkcí je označeno „zvratné se, neoddělitelné se – reflexivní tantum“. Podrobný popis viz [2], oddíl 3.3.7.5.

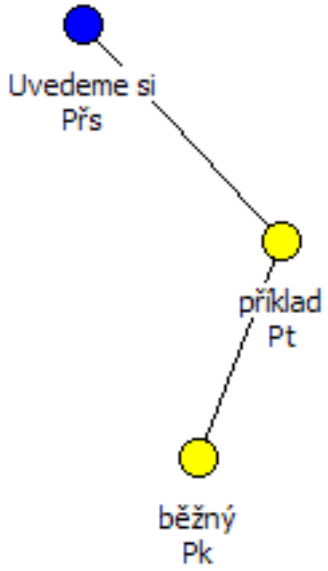
Na uzlu s funkcí *AuxT* je nutné provést operaci *připojení k rodiči*.



Obrázek 6.18: Výsledek transformace analytické funkce *AuxZ* ve školském rozboru



Obrázek 6.19: Zachycení analytické funkce *AuxO* v akademickém rozboru



Obrázek 6.20: Výsledek transformace analytické funkce *AuxO* ve školském rozboru

Příklad:
Zájem o tyto jednotky se (AuxT) *v ČR zvyšuje* (Pred) *pomalu.* (cmpr9410-049-p52s8)
Uzel *se* je připojen k uzlu *zvyšuje*. (Viz obrázky 6.21 a 6.22.)

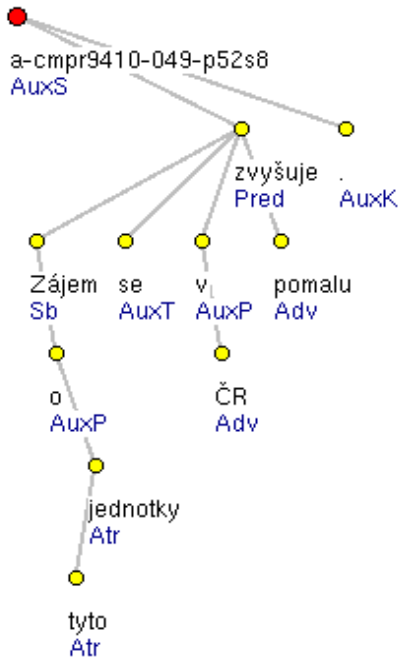
6.16 Analytická funkce *AuxR*

Touto funkcí je označeno „zvratné se, které není Obj ani AuxT (tvoří pasivum reflexivní)“. Podrobný popis viz [2], oddíl 3.3.7.6.

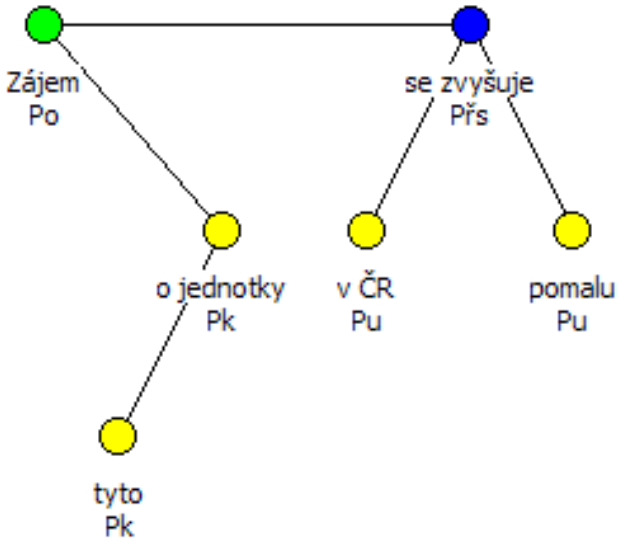
Z našeho pohledu je situace analogická s tou popisovanou v části 6.15, na uzlu s funkcí *AuxR* je nutné provést operaci *připojení k rodiči*.

6.17 Analytická funkce *AuxY*

Touto funkcí jsou označena „příslovce a částice, které nelze zařadit jinam“. Podrobný popis viz [2], oddíl 3.3.7.7.



Obrázek 6.21: Zachycení analytické funkce *AuxT* v akademickém rozboru



Obrázek 6.22: Výsledek transformace analytické funkce *AuxT* ve školském rozboru

Na uzlu s funkcí *AuxY* je nutné provést operaci *připojení k rodiči*.

6.18 Analytická funkce *AuxK*

Touto funkcí je označena „koncová interpunkce věty“. Podrobný popis viz [2], oddíl 3.3.8.2.

Koncovou interpunkci ve školské reprezentaci nezobrazujeme, proto je nutné na uzlu s funkcí *AuxK* provést operaci *odstranění uzlu*.

6.19 Analytická funkce *AuxX*

Touto funkcí je označena „čárka (ne však nositel koordinace)“. Podrobný popis viz [2], oddíl 3.3.8.3.

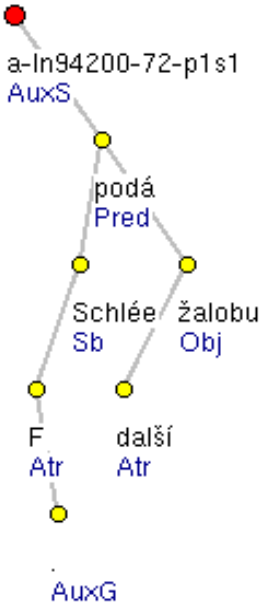
Protože jsme filtrem *SimpleSentence* z dat odstranili všechna souvětí (viz 5.2.1), uzel (čárka) s funkcí *AuxX* je nutně částí větněčlenské koordinace. Přestože mu nepřísluší ve školské reprezentaci žádný větný člen, neprovádíme na něm žádnou transformaci, uzel je totiž zpracován v rámci transformace samotného koordinačního uzlu (viz část 6.22).

6.20 Analytická funkce *AuxG*

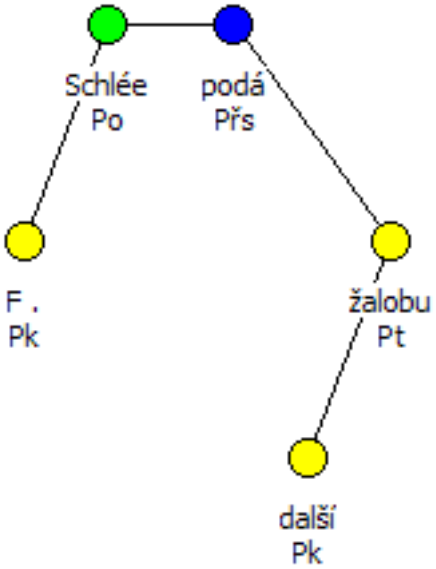
Touto funkcí jsou označeny „jiné grafické symboly, které neukončují větu“. Podrobný popis viz [2], oddíl 3.3.8.4.

Filtrem *GraphicalSymbols* (viz 5.2.2) jsme odstranili věty, v nichž existuje uzel s funkcí *AuxG*, který není tečkou. Ve zbylém případě (slovem uzlu je tečka) na uzlu provedeme operaci *připojení k rodiči*.

Příklad:
F (Atr) . (AuxG) *Schlée podá další žalobu* (ln94200-72-p1s1)
Uzel „.“ je připojen k uzlu *F*. (Viz obrázky 6.23 a 6.24.)



Obrázek 6.23: Zachycení analytické funkce *AuxG* v akademickém rozboru



Obrázek 6.24: Výsledek transformace analytické funkce *AuxG* ve školském rozboru

6.21 Analytická funkce *ExD*

Tato funkce znamená „náhradní funkci pro technické hrany vedoucí místo od elidovaného členu k ‚pseudořídícímu' slovu nebo pro hlavní člen věty bez predikátu (Ex-Dependent)". Podrobný popis viz [2], oddíl 3.4.1.

Elipsy jsme z dat vyřadili filtrem *EllipsisAposition* (viz část 5.2.3), nemusíme se jimi tedy při transformacích zabývat.

6.22 Analytická funkce *Coord*

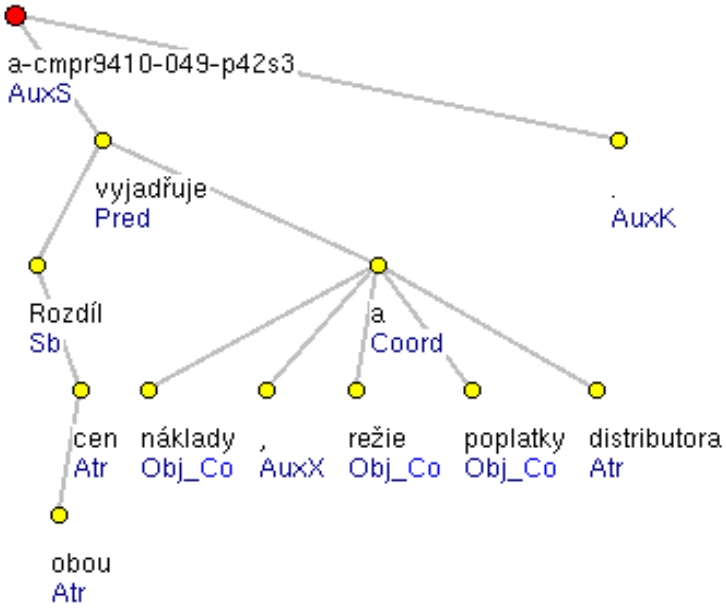
Touto funkcí je označen „koordinační uzel (souřadné spojení)". Podrobný popis viz [2], oddíl 3.5.1.

Filtrem *SimpleSentence* (viz 5.2.1) jsme z dat odstranili souvětí, proto se můžeme zabývat pouze případem, kdy uzel s funkcí *Coord* koordinuje větné členy. V této situaci je potřeba na uzlu provést operaci *pohlcení dětí*. Dojde tak především k pohlcení čárek *AuxX* a dále uzlů, jejichž analytická funkce určí, o koordinaci čeho se vlastně jedná (členem koordinace smí být pouze jeden typ uzlu, nemůže tedy být například koordinován objekt s atributem). Dále je potřeba rozebrat následující případy:

- Koordinovány jsou *Sb*. Uzel je odtržen od rodiče, zavěšen přímo pod technický uzel věty a je mu přiřazen větný člen *podmět*.
- Koordinovány jsou *Obj*, ty jsou z hlediska morfologie slovesy v infinitivu a rodičem uzlu je modální sloveso. Na uzlu je provedena operace *připojení k rodiči*.
- Koordinovány jsou *Pnom*. Dokud rodičem uzlu není uzel s funkcí *Coord* nebo *Pred*, je na uzlu prováděna operace *připojení k rodiči*.
  - Potom je-li rodičem uzlu uzel s funkcí *Coord,* je provedena ještě jedna operace *připojení k rodiči* a u rodiče je poznamenáno, že koordinuje jmennou část přísudku.
  - Jinak je provedena ještě jedna operace *připojení k rodiči* a rodiči je změněn větný člen na *přísudek jmenný*.

- Koordinovány jsou *AuxZ*. Na uzlu je provedena operace *připojení k rodiči*.
- Jinak je uzlu přiřazen ten větný člen, který odpovídá analytické funkci koordinovaných uzlů.

Příklad:
Rozdíl obou cen vyjadřuje náklady (Obj) *,* (AuxX) *režie* (Obj) *a* (Coord) *poplatky* (Obj) *distributora.* (cmpr9410-049-p42s3)
Uzel *a* pohltí uzly *náklady*, „,“, *režie* a *poplatky* a je mu přiřazen větný člen *předmět*. (Viz obrázky 6.25 a 6.26.)

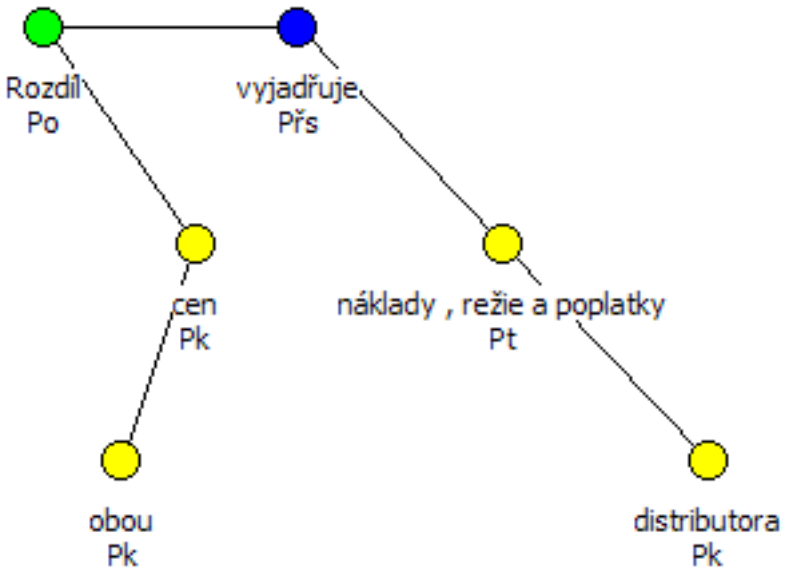


Obrázek 6.25: Zachycení analytické funkce *Coord* v akademickém rozboru

6.23 Analytická funkce *Apos*

Touto funkcí je označena „aposice (hlavní uzel)“. Podrobný popis viz [2], oddíl 3.5.2.

Aposice jsme z dat vyřadili filtrem *EllipsisAposition* (viz část 5.2.3), nemusíme se jimi tedy při transformacích zabývat.



Obrázek 6.26: Výsledek transformace analytické funkce *Coord* ve školském rozboru

Kapitola 7

Implementace

7.1 Java

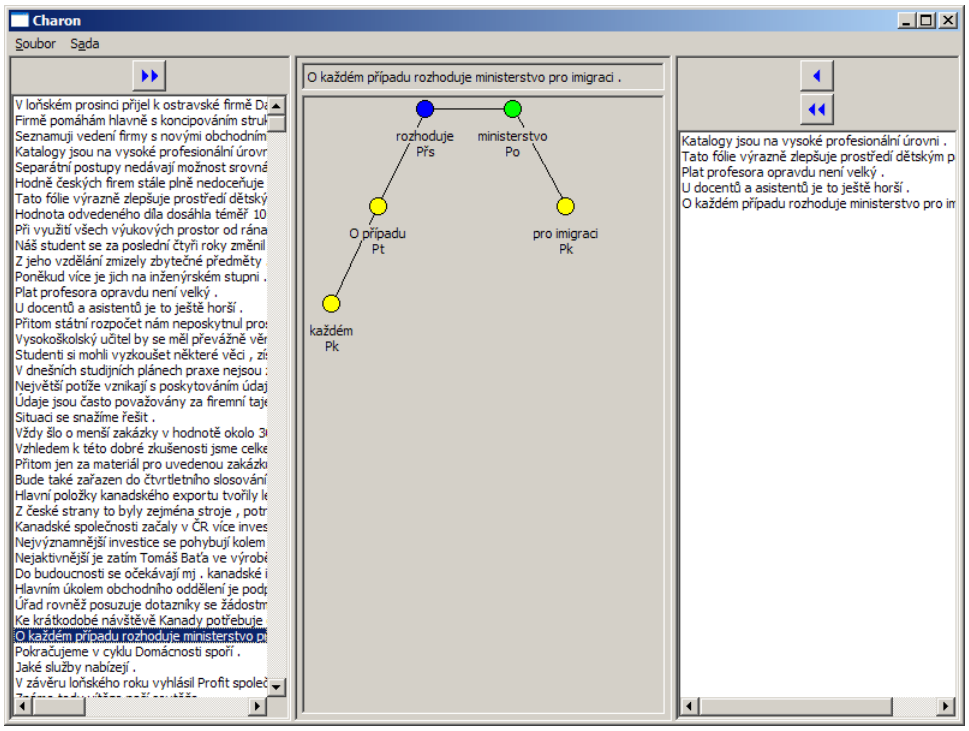
Implementaci cvičebnice (a pomocných utilit) jsme se rozhodli provést v jazyce *Java*. K tomuto rozhodnutí jsme měli několik důvodů. Nejdůležitějším asi byl ten, že Java je vysokoúrovňovým jazykem umožňujícím uživateli odpoutat se od některých pro mnohé úlohy nepodstatných detailů, navíc obsahuje řadu pojistek chránících programátora „před sebou samým“. Navíc jsme si tím zajistili možnost snadné přenositelnosti i na jiné operační systémy než MS Windows (přestože ty byly platformou, na kterou jsme se zaměřovali, z důvodu její rozšířenosti ve školách). Naopak za nevýhodu bychom mohli považovat nutnost přítomnosti JRE[1] na uživatelském počítači.

7.2 Standard Widget Toolkit

Pro práci s grafickým rozhraním byla použita knihovna SWT z projektu Eclipse[2]. Ta má oproti standardnímu modulu Swing obsaženému přímo v Javě dvě přednosti. Na každé platformě, pro kterou její implementace existuje, jsou použity standardní prvky této platformy, proto vzhled aplikace odpovídá tomu, na co je uživatel zvyklý. Navíc odezva grafického

[1] Java Runtime Environment
[2] Více viz http://www.eclipse.org/

rozhraní je i rychlejší. Následující obrázky ukazují program *Charon* spuštěný v prostředí MS Windows a GNU/Linux.

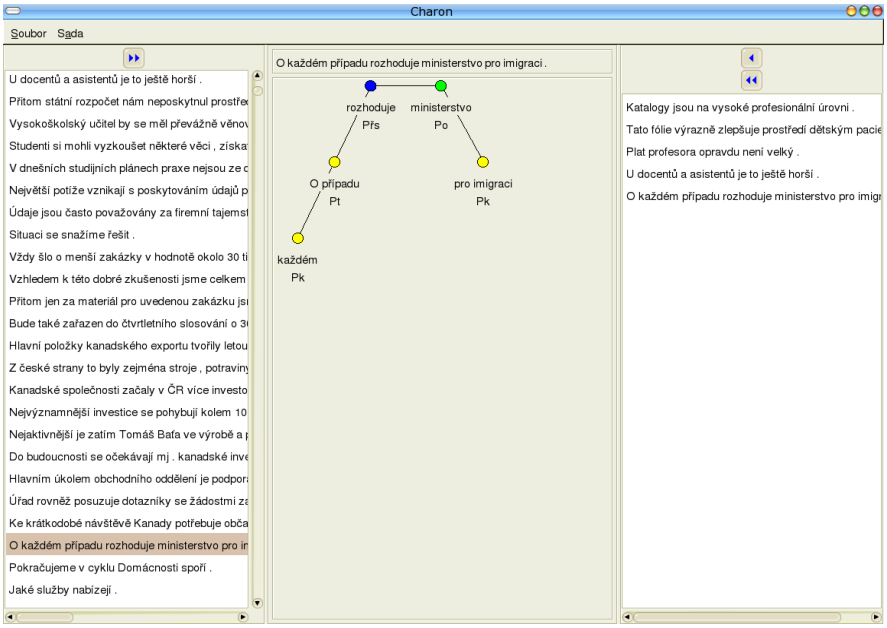


Obrázek 7.1: Program *Charon* na operačním systému MS Windows

7.3 Cvičebnice

Samotným výsledkem naší práce jsou krom tohoto textu především následující tři programy tvořící celý systém cvičebnice. Bližší informace o jejich instalaci a používání se nacházejí v *Uživatelské příručce* (dodatek A), podrobnosti z hlediska implementace jsou pak součástí *Programátorské příručky* (dodatek B).

- ***FilterSentences***. Slouží k přípravě dat vhodných k použití ve cvičebnici, koncový uživatel s ním nepřijde do styku.



Obrázek 7.2: Program *Charon* na operačním systému GNU/Linux

- ***Charon***. Administrační nástroj, slouží k zobrazování a prohlížení všech dostupných vět a vytváření cvičení. Předpokládá se, že jej bude používat vyučující.
- ***Styx***. Samotná cvičebnice, na které si budou žáci prověřovat své znalosti u cvičení vytvořených v programu *Charon*.

Kapitola 8

Závěr a výhledy do budoucna

Celý projekt cvičebnice češtiny postavené na Pražském závislostním korpusu úspěšně prošel první fází – podrobným prozkoumáním kroků, které je při zpracovávání PDT potřeba učinit, a vytvořením funkční implementace cvičebnice. Má-li však být skutečným přínosem, je potřeba v práci na něm dále pokračovat.

Dalším krokem musí být přenesení projektu z čistě akademické sféry mezi uživatele, posbírat jejich připomínky a nápady a tyto do programu zapracovat, stejně jako opravit chyby, které uživatelé naleznou. Přesto jsou některé oblasti, o kterých již nyní víme, že poskytují prostor pro celkové zlepšení programu.

8.1 Morfologie

Na morfologické úrovni je potřeba vyřešit občasnou nejednoznačnost hodnot některých kategorií v PDT (například rod u některých zájmen). Zvažujeme možnost buď'to pokusit se odvodit si tyto možnosti na základě shody, nebo nalézt všechny takovéto případy a upravit jejich morfologické značky přímo v datech.

U sloves je potřeba zapracovat načítání informací o vidu z tektogramatické roviny.

Při prezentaci projektu v rámci předmětu *Seminář z formální lingvistiky* na UK MFF vyvstala otázka, zda úlohu žákům až příliš nezjednodušujeme

tím, že za ně ke slovním druhům vybíráme patřičné morfologické kategorie (navíc pouze takové, které mají smysl, například ne všechna příslovce mají stupeň). Studenti by totiž sami měli vědět, které kategorie kdy určovat. Budeme se tedy muset zamyslet nad řešením tohoto problému.

8.2 Syntax

Z hlediska syntaktického rozboru věty musíme dokončit zobrazování doplňku. Znamená to doplnit transformaci zmíněnou v části 6.7 a dále s vyučujícími zkonzultovat nutnost kreslení druhé hrany od doplňku vedoucí.

U přívlastků a příslovečných určení je potřeba připsat načítání a zobrazování jejich druhů.

Revize bude muset doznat i zpracovávání koordinace. Zde bude situace obtížná především proto, že i jednotlivé školské učebnice se ve znázorňování několikanásobných větných členů neshodují. Navíc uváděné příklady ne vždy jasně ukazují, jak by se řešily složitější situace než ty přímo zobrazované (například rozdíl mezi rozvinutím koordinace jako celku a rozvinutím jejích jednotlivých členů). V [11] je koordinace vnímána jako jediný uzel (tedy tak, jak jsme ji pojali i my), navíc je graficky znázorněna svorkou zakreslenou nad svými členy. Autoři [13] chápou koordinaci jako jediný uzel, pokud jde o její vztah ke svému rodiči, ovšem jako několik uzlů, pokud jde o rozvíjení jejích jednotlivých členů. Naopak podle [10] obsahuje koordinace tolik uzlů, kolik má členů (a sama je pak vlastně vyjádřena pouze svorkou, zakreslenou však tentokrát pod svými členy). Ke kterým uzlům ovšem v takovém případě patří čárky a spojky, není jasné. Jak vyplývá z části 6.22, my jsme zvolili řešení, kdy koordinace je jediným uzlem, dále nedělitelným. Nevýhodou tohoto postupu je nemožnost rozlišení rozvití některých členů koordinace od rozvití celku, pokud se to ukáže při testování cvičebnice jako problém, budeme muset hledat jiné možnosti.

Podobně jako u morfologie se rovněž objevila připomínka, že i činností, kdy automaticky shlukujeme slova do jednotlivých syntaktických uzlů, studentům příliš usnadňujeme práci. I zde tedy bude nutné zkonzultovat, jak závažný problém toto je, a případně nalézt způsob, kterým studenty nechat uzly tvořit (a naopak rozebírat).

8.3 Opravy chyb

Je samozřejmé, že vytvořené programy nejsou bez chyb. Může se ukázat, že jsou potřeba další filtry, nebo že naopak některé stávající filtry odstraňují věty, které by v datovém vzorku mohly zůstat. Podobně je možné, že některé transformace nejsou prováděny vhodně či že jiné úplně chybí. Tyto všechny věci by mělo odhalit testování uživateli z řad vyučujících či žáků.

8.4 Rychlost

Rovněž na rychlosti je vždy možno zapracovat. Některé části programů jistě půjdou urychlit, jiné nikoliv, u takových by ovšem bylo vhodné uživatele nějak upozornit, že akce probíhá a že má počkat.

8.5 Konfigurovatelnost

Uživatel by měl mít možnost nějakým způsobem alespoň poněkud ovlivnit chování programu. I v tomto bodě předpokládáme, že nejvíce nápadů vzejde z testování. Například by mohl vyučující v *Charonu* nastavit, že chce do cvičebnice vybírat pouze věty obsahující (nebo naopak neobsahující) některé jevy, aby si nemusel sám hlídat, zda cvičení bude odpovídat znalostem žáků jednotlivých ročníků.

8.6 Ovládání

V tuto chvíli musí uživatel většinu akcí ve cvičebnici provádět myší. Plánujeme zaměřit se i na možnost využití různých klávesových zkratek, které by urychlovaly práci s vyplňováním cvičení.

8.7 Vnitřní implementace

Dříve, než bude moci program být šířeji využíván, bude potřeba doplnit lepší ošetřování chyb, které mohou při běhu vznikat. Z těchto důvodů existuje třída *Logger*, nicméně bude nutné ji důsledněji používat.

Rovněž plánujeme přechod na Javu verze 1.5, která nabízí několik nových užitečných vlastností. Zvažovali jsme přechod na tuto verzi již dříve, tehdy se však jednalo o novinku, u které nebylo jasné, zda neobsahuje nějaké zásadnější chyby. Navíc i uživatelé se mohli příliš čerstvý produkt zdráhat instalovat. V tuto chvíli však nám již nejsou známy důvody, proč by bylo výhodnější setrvávat u starší verze 1.4.

Dodatek A

Uživatelská příručka

Tento dokument popisuje instalaci a používání elektronické cvičebnice *Styx*.

A.1 Systémové požadavky

Cvičebnice je napsána v jazyce Java, uživatel proto musí mít nainstalováno tzv. JRE (Java Runtime Environment) verze alespoň 1.4. Zda tento požadavek splňujete, si můžete snadno ověřit. Do příkazového řádku napište `java -version`. Měla by se vám zobrazit zpráva obdobná následující:

```
java version "1.5.0_04"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_04-b05)
Java HotSpot(TM) Client VM (build 1.5.0_04-b05, mixed mode)
```

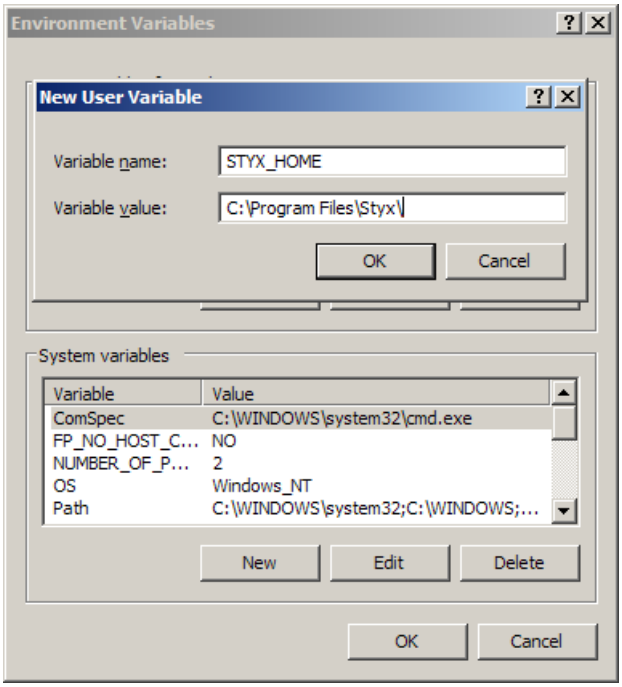
V případě, že váš systém neobsahuje Javu verze 1.4 nebo vyšší, je potřeba ji nainstalovat. Verzi pro Windows naleznete na distribučním CD ve složce *jre*.

A.2 Instalace

A.2.1 Instalace v prostředí MS Windows

1. Verze pro Windows se nachází v souboru *styx-windows.zip*. Jeho obsah rozbalte do libovolné složky v systému (např. *C:\Program Files*), na tuto složku se dále budeme odkazovat jako na *INSTALL_DIR*.

2. Tento krok by neměl být nutný na Windows postavených na jádře NT (Windows NT, 2000, XP, 2003, Vista). Nastavte systémovou proměnnou *STYX_HOME* na *INSTALL_DIR\Styx*. Podrobný postup: klikněte pravým tlačítkem myši na ikonu *Tento počítač* na ploše, z kontextového menu vyberte nabídku *Vlastnosti* (pokud nemáte na ploše ikonu *Tento počítač*, je možné dvojkliknout na ikonu *Systém* v *Ovládacích panelech*). Přepněte se na záložku *Upřesnit* a na ní stiskněte tlačítko *Systémové proměnné*. V následujícím dialogu stiskněte tlačítko *Nová* a vyplňte správné údaje (viz obrázek A.1).



Obrázek A.1: Nastavení systémové proměnné *STYX_HOME*

3. Chcete-li, vytvořte si pro program *Styx* (případně i *Charon*) zástupce, jako cíl zástupce zadejte *INSTALL_DIR\Styx\styx.bat* (respektive *INSTALL_DIR\Styx\charon.bat*).
   Tip: až si ověříte, že je cvičebnice správně nainstalována a funguje, nastavte u zástupce, aby se spouštěl v minimalizovaném okně. Vyhnete se tak viditelném probliknutí okna s příkazovým řádkem, které Windows při spuštění otvírají.

A.2.2 Instalace v prostředí GNU/Linux

1. Verze pro GNU/Linux se nachází v souboru *styx-linux.tar.bz2*. Jeho obsah rozbalte do libovolné složky v systému (např. */opt*), na tuto složku se dále budeme odkazovat jako na *INSTALL_DIR*.

2. Vytvořte si symbolický link na soubor *INSTALL_DIR/styx/styx* (případně i *INSTALL_DIR/styx/charon*) v místě, které máte součástí systémové proměnné *PATH* (např. */usr/local/bin*).

A.3 Charon

Program *Charon* slouží k prohlížení všech dostupných vět a k sestavování cvičení z těchto vět. Výběr vět je opravdu široký (přes jedenáct tisíc vět), vzhledem k jejich množství se však program může spouštět delší dobu (v závislosti na rychlosti vašeho počítače), ze stejného důvodu potřebuje k běhu volných 160 MB operační paměti.

Následující ilustrace ukazuje obrazovku programu s vyznačenými sedmi oblastmi, vysvětlíme si na něm ovládání celého programu.

1. Seznam všech vět v sadě. Slouží k označení věty (případně vět), kterou (které) chcete přidat do cvičení. Označená věta je zobrazena v oblastech 3 a 4.

2. Tlačítko sloužící k přidání označených vět do cvičení.

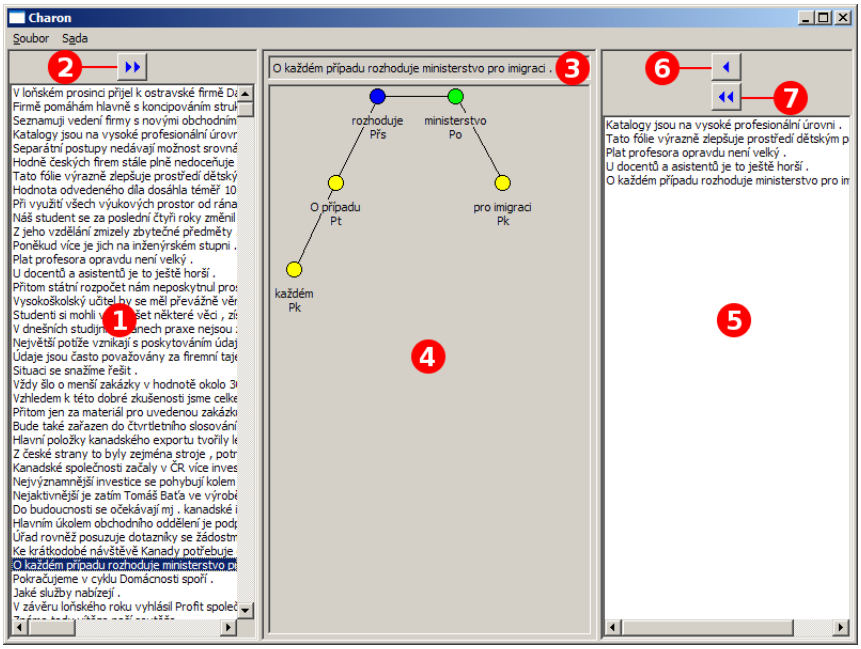
3. Zde je zobrazena označená věta. Pokud se nad některým slovem zastavíte kurzorem myši, v bublinové nápovědě jsou zobrazeny morfologické informace o tomto slově (základní tvar, slovní druh, rod, číslo, pád, ...).

4. Zde je zobrazen syntaktický rozbor věty.

5. Seznam vět dosud vybraných do cvičení. I zde je možné vybrat větu a zobrazit si ji v oblastech 3 a 4.

6. Tlačítko pro odebrání označené věty ze cvičení.

7. Tlačítko pro odebrání všech vět ze cvičení.



Obrázek A.2: Obrazovka programu *Charon*

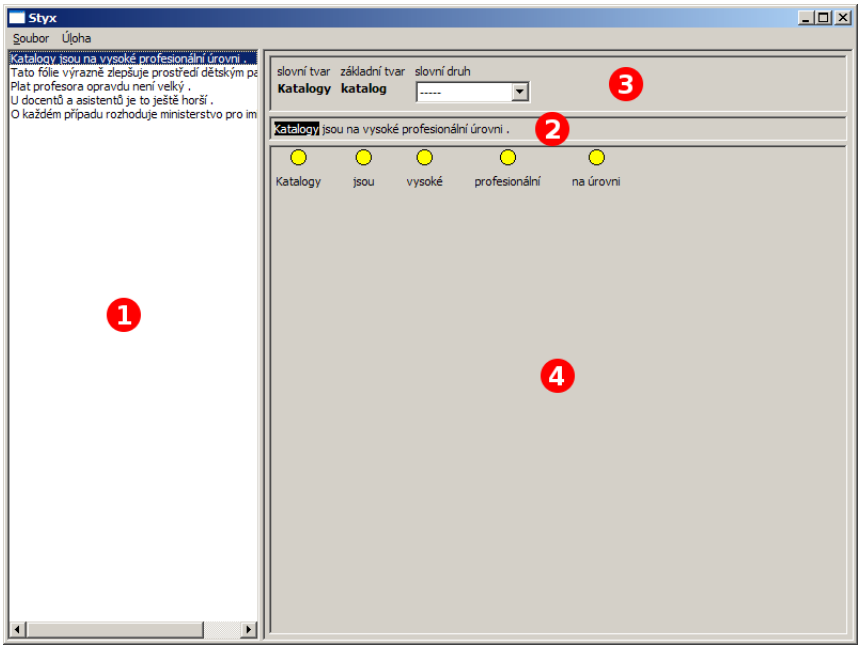
Věty, které jsou k dispozici, jsou rozděleny do deseti sad, jinak by se s programem nedalo prakticky vůbec pracovat. K přepínání sad slouží položky menu *Sada*.
Poznámka: přepnutí sady podobně jako spuštění programu (a tedy načtení první sady) nějakou dobu trvá.

Pomocí položek v menu *Soubor* si můžete cvičení uložit, nebo naopak můžete načíst cvičení dříve vytvořené a modifikovat je.

A.4 Styx

Program *Styx* je samotnou elektronickou cvičebnicí. Po spuštění se objeví víceméně prázdné okno programu, abychom mohli začít pracovat, je potřeba pomocí menu *Soubor* načíst dříve v programu *Charon* vytvořené cvičení.

Po vybrání věty, kterou si chcete zkusit rozebrat, již uvidíte situaci obdobnou následujícímu obrázku.



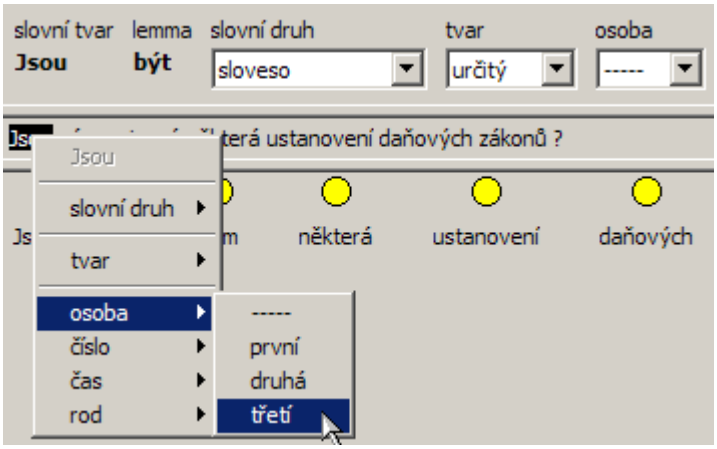
Obrázek A.3: Obrazovka programu *Styx*

1. Seznam vět ve cvičení. Slouží k označení věty, kterou chcete zpracovávat.
2. Zde je zobrazena vybraná věta.
3. Oblast pro určování tvarosloví.
4. Oblast pro syntaktický rozbor věty.

A.4.1 Určování morfologie

Nejprve je potřeba vybrat slovo, u kterého chcete morfologii určovat – klikněte na něj v oblasti 2 (viz obrázek A.3). Nyní můžete zvolit jeho slovní druh pomocí roletky v oblasti 3. Po zvolení slovního druhu se objeví další roletky pro výběr morfologických kategoriích slovnímu druhu příslušejících.

Alternativně je možno hodnoty tvarosloví vybírat z kontextových menu vyvolaných pravým tlačítkem myši nad zvoleným slovem (viz obrázek A.4).



Obrázek A.4: Kontextové menu pro výběr morfologických kategorií

A.4.2 Syntaktický rozbor věty

Stavba syntaktického stromečku probíhá metodou *drag and drop*. Uchopte myší uzel za kolečko, přetáhněte jej nad uzel, který se má stát jeho rodičem a tam jej upusťte.

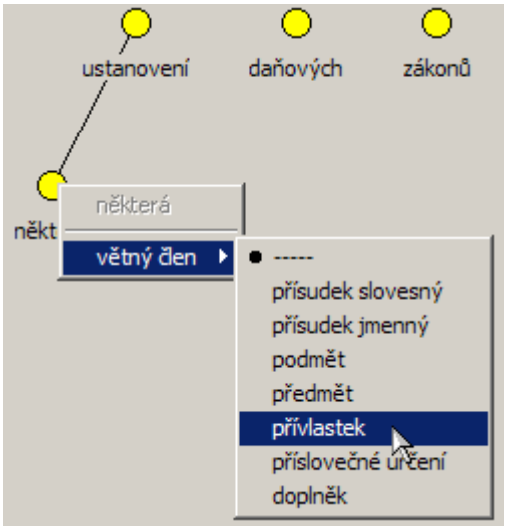
Větné členy uzlům přiřadíte pomocí kontextového menu vyvolaného nad uzlem (viz obrázek A.5). Po označení podmětu i přísudku dojde automaticky k jejich spojení do základní skladební dvojice.

A.4.3 Kontrola cvičení

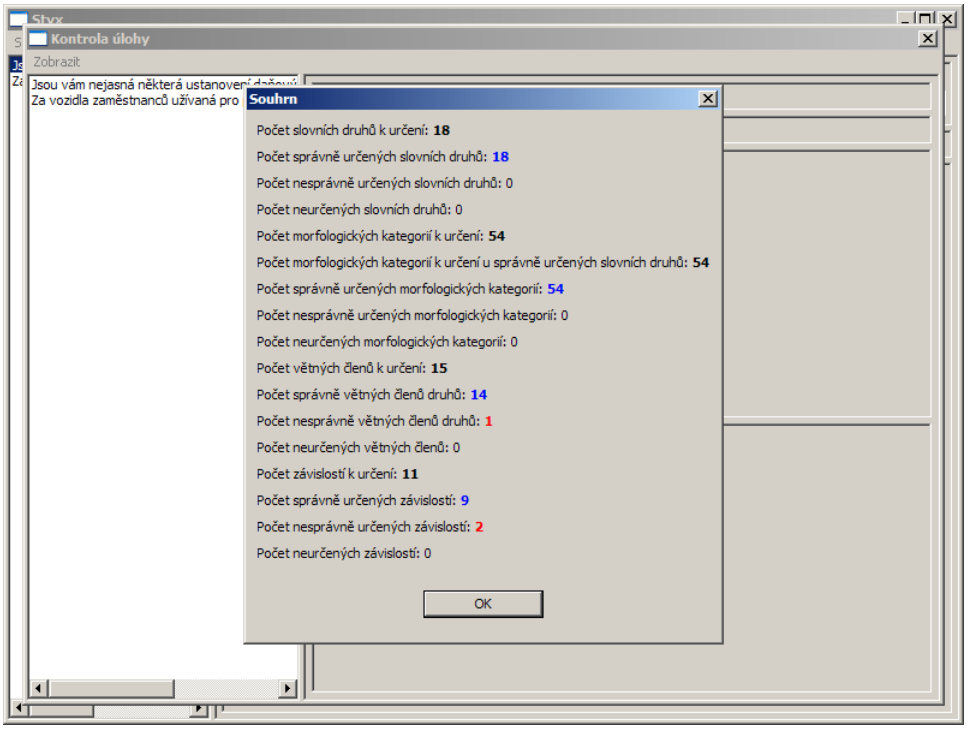
Jste-li s řešením cvičení hotovi, můžete si zkontrolovat výsledky pomocí volby *Zkontrolovat* v menu *Úloha*. Nejprve se vám zobrazí celková statistika správně a špatně určených částí celého cvičení (viz obrázek A.6). Po jejím potvrzení (lze ji kdykoliv později znovu vyvolat) se můžete podívat podrobně na jednotlivé správné a chybné odpovědi (viz obrázek A.7).

1. Seznam vět ve cvičení. Slouží k označení věty, kterou si chcete zkontrolovat.

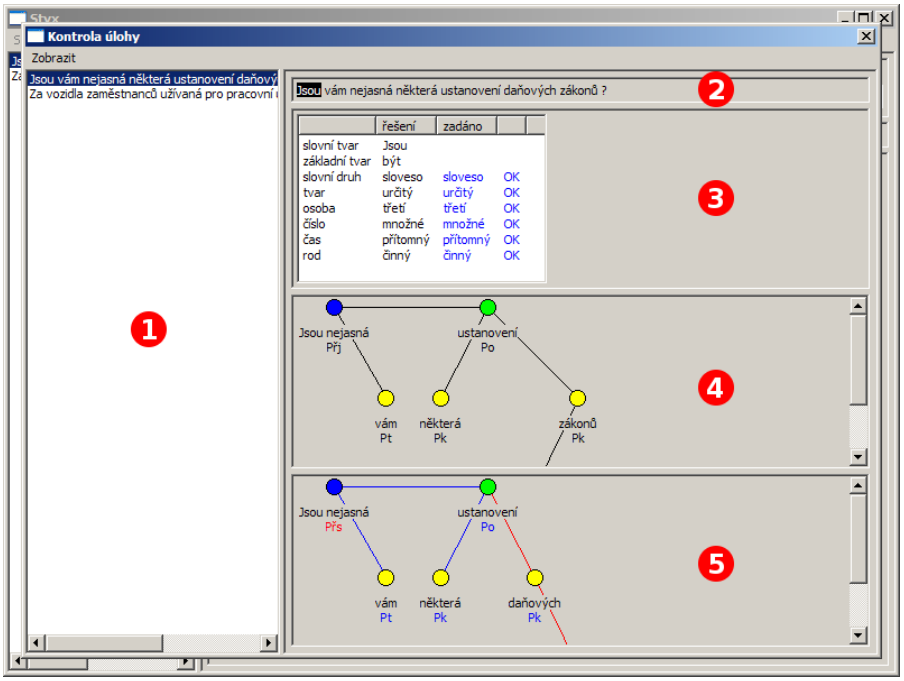
2. Zde je zobrazena vybraná věta. Kliknutím na slovo zobrazíte informace o morfologii v oblasti 3.



Obrázek A.5: Kontextové menu pro výběr větného členu



Obrázek A.6: Kontrola cvičení – souhrn



Obrázek A.7: Kontrola cvičení

3. Zde je zobrazena tabulka shrnující tvarosloví vybraného slova. Pokud jste danou morfologickou kategorii určili správně, je zobrazena modře, pokud špatně, je zobrazena červeně.

4. Zde je zobrazen vzorový syntaktický rozbor věty.

5. Zde je zobrazen váš rozbor věty. Správně určené závislosti a větné členy jsou zobrazeny modře, špatně určené červeně.

Dodatek B

Programátorská příručka

B.1 Příprava potřebných souborů

V následujícím textu se bude častokrát předpokládat, že máme z distribučního CD někam zkopírované a připravené potřebné soubory. Především jsou to data PDT, která jsou navíc na CD kvůli velikosti komprimována a je potřeba je tedy rozbalit. Dále se jedná o adresář *src* obsahující veškeré zdrojové kódy a soubory *Makefile* (ten řídí kompilaci *Styxu* a *Charonu*, program *FilterSentences* má svůj vlastní *Makefile* v *src/utils/prepare*), *StyxManifest* a *CharonManifest*. (Pro kompilaci na GNU/Linuxu místo nich použijte *Makefile.linux*, *StyxManifest.linux* a *CharonManifest.linux*.)

B.2 Program FilterSentences

Program *FilterSentences* slouží k filtrování vět Pražského závislostního korpusu (blíže o filtrování pojednáváme v kapitole 5). S programem nikdy nebude pracovat koncový uživatel, proto jeho popis je zde a nikoliv v uživatelské příručce. Spuštění programu má následující formu:

```
java <java_parameters> utils.prepare.FilterSentences
-f <filter> [-o <output_prefix>] file [file2 ...]
```

Povinnými parametry jsou alias filtru a vstupní soubor. Aliasy filtrů jsou definovány přímo ve třídě *FilterSentences*, jsou to řetězce *sf001*, *sf002*, ...

Jako vstupní soubor musí být udán soubor obsahující tektogramatické informace PDT (ostatní tři soubory jsou dohledány automaticky, předpokládá se, že jejich názvy vzniknou záměnou přípony *.t* za *.w*, *.m*, respektive *.a*). Výstupní soubory jsou vytvořeny v místě vstupních, jejich jména jsou odvozena od vstupních souborů tak, že před příponu je ještě přidán řetězec *.alias* podle zvoleného aliasu filtru.

Parametr *-o* způsobí, že věty ze všech vstupních souborů budou zapsány do jediného výstupního souboru (přesněji do jediné čtveřice výstupních souborů). Tento parametr musí být na příkazové řádce uveden až za parametrem *-f*. Řetězec *output_prefix* určuje název (a umístění) výstupních čtyř souborů, je doplněn o patřičné přípony.

B.2.1 Průběh filtrování

V této části vysvětlíme, jakými konkrétními příkazy jsme filtrování prováděli. Protože možnosti skriptování z příkazového řádku na MS Windows jsou poměrně omezené, vypomáhali jsme si portem unixovských utilit *Cygwin*[1]. Před spuštěním následujících příkazů se předpokládá, že máme připraveno vše potřebné (viz B.1).

Z adresáře *src/utils/prepare* spustíme následující příkaz:

```
java -classpath
"../..;../../../lib/xercesImpl.jar;../../../lib/xml-apis.jar"
utils.prepare.FilterSentences -f sf001 d:/pdt/dtest/*.t
```

Analogicky postupujeme s ostatními adresáři. Program *FilterSentences* vytváří v aktuálním adresáři soubor *debug.log*, ve kterém se mimo jiné nachází informace, kolik vět bylo zachováno. Máme-li první filtr aplikován na všechna data, můžeme spustit druhý:

```
java -classpath
"../..;../../../lib/xercesImpl.jar;../../../lib/xml-apis.jar"
utils.prepare.FilterSentences -f sf002 d:/pdt/dtest/*.sf001.t
```

Takto pokračujeme se všemi filtry kromě posledního. Výsledně totiž nebudeme chtít pracovat se stovkami jednotlivých souborů, proto poslední filtr aplikujeme takto:

[1] Viz http://www.cygwin.com/

```
java -classpath
"../..;../../../lib/xercesImpl.jar;../../../lib/xml-apis.jar"
utils.prepare.FilterSentences -f sf008 -o d:/pdt/dtest
d:/pdt/dtest/*.sf007.t
```

Tak vytvoříme soubory *dtest.w*, *dtest.m*, *dtest.a* a *dtest.t* obsahující všechny věty z celého adresáře *dtest*, které filtry prošly.

B.3 Generovaná dokumentace

Ze zdrojových kódů je automaticky generována dokumentace programem *javadoc*. Tato dokumentace se nachází na distribučním CD v adresáři *doc/javadoc*. Je dobrým zdrojem dalších informací o všech níže zmiňovaných třídách, nebudeme to proto u nich jednotlivě zdůrazňovat.

B.4 Formát dat Styxu a Charonu

Soubor se zdrojovými daty pro Charon se jmenuje *data.styx* (a je obsažen v souboru *styx-data.jar*, blíže viz B.13). Jedná se o obyčejný soubor ve formátu ZIP, který obsahuje zkomprimované soubory *dtest.w*, *dtest.m*, ..., *train-8.t*, které jsme vytvořili během filtrování. Uživatelem v Charonu vytvořená cvičení jsou opět soubory ve formátu ZIP, které obsahují zkomprimované soubory *styx.w*, *styx.m*, *styx.a* a *styx.t* – jejich obsahem jsou do cvičení vybrané věty ve formátu PML.

B.5 Package utils.prepare

Obsahuje především třídy přímo se týkající programu *FilterSentences*. Jedná se o třídy *FilterSentences* (samotný program), *SimpleSentence*, *GraphicalSymbols*, *EllipsisApposition*, *OnePredicate*, *LessThanNWords*, *MoreThenNWords*, *AuxO*, *IndividualSentences*, *KeepAll* (jednotlivé filtry) a rozhraní *SentenceSelector*.

B.5.1 SentenceSelector

Toto rozhraní musí splňovat každé filtrační kritérium. Obsahuje jedinou metodu, *keepSentence*, jejímiž parametry jsou informace o větě na všech čtyřech rovinách a jejíž návratová hodnota určuje, zda má být věta zařazena do výstupního souboru vět.

B.5.2 DOMUtils

Tato třída obsahuje metody pro usnadnění několika typických manipulací s DOM stromem XML dokumentu, které je třeba často při filtrování provádět. Je však využívána nejen při filtrování, ale i programy *Styx* a *Charon*.

B.5.3 ElementList

Tato třída je jednoduchou implementací rozhraní *org.w3c.dom.NodeList*.

B.6 Package styx.gui

Obsahuje třídy poskytující *Styxu* a *Charonu* metody pro práci s grafickým rozhraním. Třída *GUIController* řídí prakticky vše, co se v programech na obrazovce objevuje, třída *TaskChecker* obhospodařuje okno se zobrazením kontroly cvičení. Třídy *FontProvider* a *ColorProvider* se starají o práci s fonty a barvami, především o jejich vytvoření, udržování v cache a posléze navrácení operačnímu systému, když už nejsou potřeba.

B.6.1 StyxNodeComposite

StyxNodeComposite je tzv. custom SWT komponenta, graficky reprezentuje třídu *StyxNode* na obrazovce (viz B.8.5). Vizuálně obsahuje kolečko, za které je možno ji přesouvat (případně ji připojovat k jiné), a dva textové popisky: seznam slov a analytickou funkci asociované instance třídy *StyxNode*. Styl předávaný konstruktoru určuje, zda může být komponenta modifikovatelná či zda se pro zobrazování mají použít hodnoty z dat, nebo hodnoty zadané uživatelem.

B.7 Package styx.i18n

Obsahuje třídu *MessageProvider* zodpovědnou za načítání veškerých textů zobrazovaných uživateli. Ta umožňuje snadný překlad celého grafického rozhraní do jiného jazyka.

B.8 Package styx.linguistics

Obsahuje třídy nějakým způsobem související s lingvistickým pozadím celé úlohy. Třída *ModalVerbs* obsahuje seznam modálních sloves a poskytuje metody pro zjištění, zda dané slovo je modálním slovesem. (Pozor: zdrojový kód této třídy je generován teprve při překladu pomocí programu *native2ascii*.) Třída *PMLConstants* obsahuje konstanty související s načítáním dat ve formátu PML. Třída *TagUtils* obsahuje konstanty pro hodnoty jednotlivých morfologických kategorií, stejně jako struktury určující, jaké kategorie patří kterému slovnímu druhu (a v jakém pořadí).

B.8.1 PMLFile

Tato třída reprezentuje jednu čtveřici souborů ve formátu PML. Obsahuje metodu *parse* umožňující načíst ze souborů veškeré potřebné informace o větách, které obsahují.

B.8.2 PMLSentences

Tato třída reprezentuje množinu vět, kterou poskytuje buď jako pole nebo jako mapu (u které jsou klíči identifikátory vět).

B.8.3 PMLSentence

Tato třída reprezentuje jednu větu. Obsahuje množinu slov, kterou poskytuje buď jako pole nebo jako mapu (u které jsou klíči identifikátory slov

na analytické rovině). Navíc umožňuje na svých slovech provést transformace, o kterých hovoříme v kapitole 6, a vytvořit tak reprezentaci odpovídající školské výuce. Přesněji je touto reprezentací strom instancí třídy *StyxNode* (viz B.8.5).

B.8.4 PMLWord

Tato třída reprezentuje jedno slovo. Obsahuje položky (a jejich *get* a *set* metody) odpovídající potřebným údajům o slově z morfologické a analytické roviny PDT.

B.8.5 StyxNode

Tato třída reprezentuje jeden uzel na školské syntaktické rovině. Každý uzel obsahuje jedno či více slov a má právě jednoho rodiče (s výjimkou kořene) a žádné nebo několik dětí. Tak tvoří uzly patřící do jedné věty syntaktický strom. Jeho kořen má maximálně dvě děti – podmět a přísudek. Uzel dále obsahuje větný člen, navíc jedno z jeho slov je zvoleno jako jeho tzv. *hlava*, je to právě to slovo, z jehož analytické funkce uzel získal svůj větný člen.

B.9 Package styx.log

Obsahuje třídu *Logger*. Ta poskytuje aplikační a ladící log a metody, pomocí kterých do nich lze zapisovat.

B.10 Package styx.resource

Obsahuje třídu *ResourceProvider*, která je zodpovědná za načítání různých datových zdrojů jako například obrázků.

B.11 Package styx

Obsahuje samotné programy *Styx* a *Charon*.

B.12 Překlad

Překlad programů řídí soubor *Makefile*. Jeho důležité cíle:

- *clean*: smaže při překladu vzniklé soubory
- *all*: provede kompilaci
- *rebuild*: provede rekompilaci, tedy nejprve provede *clean* a poté *all*
- *jars*: vytvoří distribuční soubory ve formátu JAR
- *javadoc:* vygeneruje dokumentaci ze zdrojových kódů.

B.13 Obsah JAR souborů

Při překladu vzniká pět distribučních souborů: *styx-data.jar* obsahuje zdrojová data pro Charon, tedy soubor *data.styx*, *styx-resources.jar* obsahuje datové zdroje (například obrázky), *styx-common.jar* obsahuje společné třídy pro Styx i Charon, *styx.jar* obsahuje samotný program Styx a *charon.jar* obsahuje samotný program Charon.

Dodatek C

Rejstřík obrázků a tabulek

Na dalších stránkách se nachází seznam obrázků a tabulek, které jsou v textu použity.

Seznam obrázků

Seznam tabulek

Literatura

[1] Pražský závislostní korpus. http://ufal.mff.cuni.cz/pdt2.0/.

[2] Jan Hajič, Jarmila Panevová, Eva Buráňová, Zdeňka Urešová, Alla Bémová. *Anotace Pražského závislostního korpusu na analytické rovině: pokyny pro anotátory*, 1999.

[3] Jiří Hana, Daniel Zeman. *Anotace Pražského závislostního korpusu na morfologické rovině: pokyny pro anotátory*, 2005.

[4] Barbora Hladká, Ondřej Kučera. Prague Dependency Treebank as an exercise book of Czech. *Proceedings of HTL/EMNLP 2005 Interactive Demonstrations, Vancouver, BC, Canada*, 2005.

[5] Marie Mikulová, Allevtina Bémová, Jan Hajič, Eva Hajičová, Jiří Havelka, Veronika Kolářová, Lucie Kučová, Markéta Lopatková, Petr Pajas, Jarmila Panevová, Magda Razímová, Petr Sgall, Jan Štěpánek, Zdeňka Urešová, Kateřina Veselá, Zdeněk Žabokrtský. *Anotace Pražského závislostního korpusu na tektogramatické rovině: pokyny pro anotátory*, 2005.

[6] Petr Pajas. Tree Editor TrEd. http://ufal.mff.cuni.cz/∼pajas/tred/, 2000–2005. Software.

[7] Petr Pajas, Jan Štěpánek. *Generic XML-Based Format for Structured Linguistic Annotation and Its Application to Prague Dependency Treebank 2.0*. Technická zpráva č. 29, ÚFAL MFF UK, 2005.

[8] Silcom Multimedia. Didakta Český jazyk 1. Software.

[9] Pavel Srp, Ivana Sklenářová, Martina Fritschová. Český jazyk, přijímací zkoušky na SŠ, 2004. Software.

[10] Vlastimil Styblík, Marie Čechová, Přemysl Hauser, Eva Hošnová. *Český jazyk pro 7. ročník základní školy a pro odpovídající ročník víceletých gymnázií*. SPN – pedagogické nakladatelství, 2002.

[11] Vlastimil Styblík, Marie Čechová, Přemysl Hauser, Alois Jedlička, Bohumil Sedláček. *Český jazyk pro 7. ročník základní školy*. Fortuna, 1995.

[12] Terasoft, a. s. TS Český jazyk 2 – jazykové rozbory, 2003. Software.

[13] Zdeněk Topil, Vladimíra Bičíková. *Český jazyk s Tobiášem, Skladba – Věta jednoduchá I.* Tobiáš, 1996.

[14] Lubomír Šára, David Šára. PON Škola – Český jazyk, 2003. Software.