

MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY



Extrakce strukturních informací z běžného textu na základě syntaktického analyzátoru

BAKALÁŘSKÁ PRÁCE

Miloš Jakubíček

Brno, 2008

Prohlášení

Prohlašuji, že tato bakalářská práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Vedoucí práce: Aleš Horák, Ph.D.

Poděkování

Rád bych na tomto místě poděkoval Aleši Horákovi, Ph.D. za ochotu, trpělivost a pečlivost při odborném vedení této práce, kolegům v laboratoři zpracování přirozeného jazyka (zvláště pak Vladimíru Kadlecovi a Vojtěchovi Kovářovi) za cenné podněty týkající se implementace a v neposlední řadě rodině a přítelkyni za podporu a potřebné tvůrčí zázemí.

Shrnutí

Práce popisuje využití syntaktického analyzátoru **synt**, vyvíjeného v Laboratoři zpracování přirozeného jazyka FI MU, k získání informací o syntaktických strukturách běžných vět v českém jazyce (jako jsou např. jmenné či slovesné skupiny), které z pohledu syntaktické analýzy zpravidla odpovídají netermínálům v gramatice, na základě které hledá analyzátor možná platná odvození zadané věty. Tento syntaktický analyzátor byl rozšířen tak, aby stávající výstup využil k extrakci těchto syntaktických struktur a umožnil různým způsobem jejich identifikaci. Za tímto účelem byly v analyzátoru rozpracovány i některé dosud nevyužité výsledky syntaktické analýzy, jež umožňují další zpřesnění morfologické analýzy, a tím i větší rozčlenění syntaktických (pod)struktur.

Abstract

This thesis describes the utilization of syntactic parser **synt** (developed in the Natural language processing laboratory at FI MU) to gain information about syntactic structures (such as noun or verb phrases) of common sentences in Czech. These structures are from the analysis point of view usually identical non-terminals in the grammar used by the parser to be searched for possible valid derivations of given sentence. This syntactic parser has been extended in such a way that enables the current output to be used to extracting those syntactic structures and give several ways how to identify them. To achieve this, some previously unused results of syntactic analysis have been evolved which led to more accurately morphological analysis and hence also deeper distinction among various syntactic (sub)structures.

Klíčová slova

syntaktická analýza, syntaktické struktury, morfologická analýza, desambiguace, extrakce struktur, zpracování přirozeného jazyka

Keywords

syntactic analysis, parsing, syntactic structures, morphological analysis, desambiguation, extraction of structures, natural language processing

Obsah

Úvod	2
1 Lingvistická východiska	3
1.1 <i>Počítačové zpracování přirozeného jazyka</i>	3
1.2 <i>Nejednoznačnost v přirozeném jazyce</i>	4
2 Syntaktický analyzátor synt	7
3 Extrakce syntaktických struktur	10
3.1 <i>Syntaktické struktury</i>	10
3.2 <i>Zpětné zpřesnění výsledků morfologické analýzy</i>	14
3.3 <i>Faktory ovlivňující zpětnou propagaci</i>	16
3.3.1 <i>Rozlišovací schopnost gramatiky</i>	16
3.3.2 <i>Morfologické marginálie</i>	18
4 Implementace	20
4.1 <i>Průchod chartem</i>	20
4.2 <i>Zpětná propagace výsledků morfologické analýzy</i>	23
4.3 <i>Extrakce struktur</i>	26
4.3.1 <i>Algoritmy</i>	26
4.3.2 <i>Datové struktury</i>	27
5 Další vývoj	28
Závěr	29
Literatura	31
A Příklad výsledného chartu	32
B Úplný výčet použitých funkcí	33
C Volby syntu relevantní pro extrakci struktur	36

Úvod

Počítačové (strojové) zpracování přirozeného jazyka zahrnuje širokou škálu problémů, které se dotýkají různých úrovní toho, co lze souhrnně nazvat přirozeným jazykem. Jedná se zejména o následující úkoly:

- **analýzu a syntézu řeči**, tj. převod mluvené řeči na text a obráceně,
- **analýzu a syntézu přirozeného jazyka v textové podobě**.

Alespoň částečně úspěšné zvládnutí těchto dvou problémů je základním předpokladem pro mnoho dalších úkolů vyššího řádu, jako je extrakce informací, strojový překlad, implementace inteligentních dialogových systémů nebo logická analýza přirozeného jazyka – obecně tedy *porozumění* přirozenému jazyku.

Cíle jazykového „inženýrství“, ve dvou větách a přesto – alespoň dle mého soudu – velmi výstižně, vyjádřil v nedávné době doc. Karel Pala: „Naší snahou je učinit komunikaci mezi člověkem a strojem dvoucestnou. Chceme, aby nejen člověk rozuměl stroji, ale hlavně aby počítače rozuměly člověku podstatně lépe než dosud.“ [Pal08, s. 4]

Kapitola 1

Lingvistická východiska

1.1 Počítačové zpracování přirozeného jazyka

Nejen z pohledu lingvistiky, ale většinou i při praktických implementacích různých systémů je zpracování přirozeného jazyka rozlišováno podle jednotlivých jazykových *rovin* tvořících následující hierarchii¹:

- **fonetické**, zahrnující analýzu a syntézu řeči,
- **morfologické**, zkoumající tvary slov a jejich částí,
- **syntaktické**, zabývající se skladbou vět a jejich částí,
- **sémantické**, tj. významové,
- **pragmatické**, hledající význam v kontextu úmyslu mluvčího, místa promluvy a času apod.,
- **(logické)**², využívající vhodného logického systému pro logickou analýzu přirozeného jazyka, jakým je například TIL.³

Vztah informatiky k přirozenému jazyku je velmi úzký, a to nejen z hlediska přínosu úspěšných aplikací, ale i z pohledu teoretické informatiky, jejíž základy staví mj. na teorii formálních (tedy uměle vytvořených) jazyků. Narozdíl od formálních jazyků, jejichž systém je za desítky let velmi podrobně prozkoumán a které vznikly právě za účelem strojového zpracování, představuje zpracování přirozeného jazyka v mnoha ohledech fundamentálně odlišný problém s celou řadou zcela netriviálních úkolů, jejichž řešení je stále v nedohlednu.

Živelně vznikající a neustále se měnící přirozený jazyk totiž klade svému strojovému zpracování mnoho překážek pramenících z podstaty toho, jak vzniká a jakým způsobem je používán, za zmínku stojí především:

-
1. Srovnej [PO92, s. 43] [PMZ89, s. 24–25] [Hau99, s. 18–21].
 2. Záměrně uvádím pouze v závorkách, protože logika jako jev striktně *mimojazykový* nemůže tvořit jazykovou rovinu. V kontextu jazykové analýzy je ovšem dalším přirozeným krokem v komunikaci člověka s počítačem.
 3. Blíže např. [PMZ89] a [Hor01, s. 36 a následující].

- **nesystematičnost** projevující se např. značným výskytem výjimek či obtížně kategorizovatelných jevů,
- **masivní nejednoznačnost**, omezíme-li se vždy jen na některou jazykovou rovinu či roviny,
- **složitost** ústící v nutnost použití komplikovaných formalismů, které s sebou přináší řadu dalších problémů (silný formalismus vs. časová a prostorová výpočetní proveditelnost, vyjadřovací síla formalismu vs. snadnost jeho uchopení a přímočarost abstrakce apod.),
- **jazyková závislost**, vinou které je v řadě případů rozšíření výsledků na další jazyky velmi obtížné.

1.2 Nejednoznačnost v přirozeném jazyce

Pod nejednoznačností přirozeného jazyka nemám na mysli ani tak vzájemné komunikační nepochopení⁴, jako spíše nejednoznačnost na úrovni jednotlivých jazykových rovin. Tento druh jazykové víceznačnosti si často při běžné komunikaci ani neuvědomujeme, protože bleskové zpracování jazyka (v psané i mluvené podobě) naším mozkem, nejexpresivnějším výpočetním nástrojem, nás většinou nenechá na pochybách o významu sdělení.

Jen stěží lze usuzovat, že by náš mozek pracoval podobně primitivním způsobem, jako je k tomu okolnostmi nucen počítač, tedy analyzoval postupně zvlášť po jednotlivých jazykových rovinách⁵. Právě proto okamžitě nevidíme případné jazykové víceznačnosti vyskytující se v nižších jazykových rovinách, než v jakých jsme zvyklí uvažovat – tedy zejména v *pragmatických*. Při počítačovém zpracování přirozeného jazyka však tento jev představuje závažný problém, jehož překonání nelze automaticky předpokládat se zvyšující výpočetní silou (či obecně s rostoucími kvantitativními parametry) dnešních počítačů. Nejednoznačnost je třeba většinou řešit zvlášť případ od případu (na jednotlivých jazykových úrovních a i v rámci nich podle různých dílčích typů) jedná se o podstatný faktor, který značným způsobem znehodnocuje výsledky analýzy.

4. Přestože nelze popřít, že k němu dochází často a leckdy může představovat závažný problém, z podstaty této úlohy nelze v brzké budoucnosti očekávat jeho strojové řešení.

5. Jakkoli např. alespoň částečně oddělené zpracování syntaxe a sémantiky v Brocově a Wernickeho mozkovém centru se má za prokázané [Kou05, s. 86–87].

Níže uvádím pro názornost (ovšemže nikoli vyčerpávající!) příklady jazykové víceznačnosti společně s jevy, na kterých jsou tyto příklady založeny, rozlišené podle jednotlivých jazykových rovin; zčásti se jedná o příklady vlastní, zčásti převzaté z různých zdrojů dle příslušných odkazů:

- **fonetika:** zánik a spodoba znělosti: [standartpi:]⁶ – standardní nebo standartní?
- **morfologie:** nesklonná slova (podle vzoru „želé“), slova s totožným pádovým tvarem, ale různými lemmaty: „myslím“, ...
- **syntax:** volný pořádek slov ve větě, aktuální větné členění: „Auta předjela letadla“ – kdo koho předjel?
- **sémantika:** homonyma: „*Rys* ležel opodál.“, anaforické výrazy: „*Jakmile* uviděl chlapce s jedoucím kočárkem, *hned ho* zastavil.“

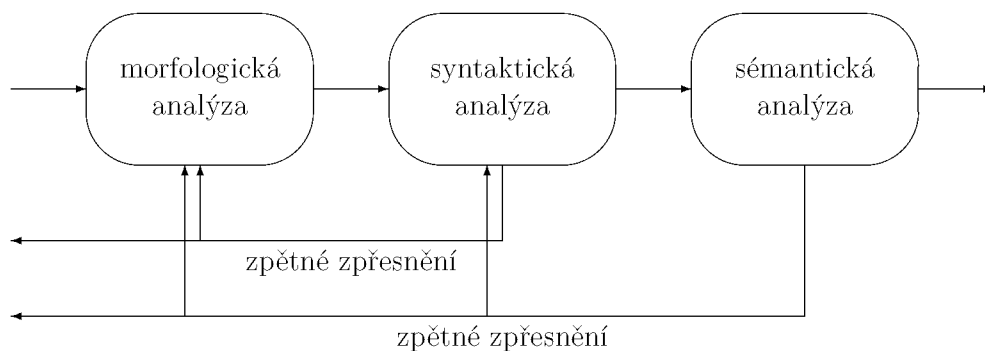
Na výše uvedených příkladech je dobře patrný základní rys většiny jazykových víceznačností: pramení z toho, že nejsme schopni určit význam jednoznačně s využitím *pouze* znalostí příslušné jazykové roviny. Následná analýza na vyšší jazykové úrovni však v některých případech může vést k jednoznačnému určení jazykových atributů nižší úrovně. Jestliže u samotného slova „myslím“ nejsme schopni odlišit⁷, zda se jedná o dativ, resp. instrumentál plurálu substantiva „mysl“ nebo indikativ přítomného aktiva slovesa „myslet“ v 1. osobě, syntaktická analýza bude schopná na tuto otázku zpravidla jednoznačně odpovědět. Obdobně pro rozlišení sémantických víceznačností může pomoci znalost kontextu dané promluvy („*Rys ležel opodál. Spokojeně odpočíval.*“).

Vyvstává proto otázka, zda by nebylo vhodnější analyzovat přirozený jazyk ve všech jazykových rovinách současně nebo přinejmenším takovým způsobem, který by umožňoval současnou analýzu několika rovin naráz. Toto řešení ovšem naráží na zcela zásadní překážku: aby bylo možné provést analýzu v určité jazykové rovině, je zpravidla nezbytná – byť částečná a nepřesná – analýza všech rovin předcházejících. Například není možné provádět syntaktickou analýzu bez předchozí znalosti *alespoň potenciálních* morfologických atributů jednotlivých slov ve větě, pouze na základě povědomí o znacích, bez dalších informací o slovech a jejich elementárních slovnědruhových charakteristikách.

6. Pro fonetický zápis slov je použita Mezinárodní fonetická abeceda IPA (International Phonetic Alphabet), blíže viz <http://www.arts.gla.ac.uk/IPA>.

7. Přinejmenším ne spolehlivě, jakkoli statistické zhodnocení obou možností v tomto konkrétním případě dává jednoznačný (a správný) výsledek.

V praxi lze tedy spíše uvažovat postup, při kterém dochází k postupnému zpřesňování výsledků analýzy během jejího postupného provádění od nejnižších po nejvyšší jazykové úrovně, jak ilustruje schéma na následujícím obrázku č. 1.1.



Obrázek 1.1: Postupné zpřesňování analýzy při zpětném použití později dosažených výsledků.

Jedním z příkladů užití tohoto postupu je i sekundární výsledek této práce, a sice využití výsledků syntaktické analýzy ke zpřesnění morfologické analýzy vstupní věty (přesněji řečeno k vyloučení některých morfologických atributů, které by se bez znalosti syntaxe jevily jako potenciálně možné) – více viz podkapitolu 3.2 Zpětné zpřesnění výsledků morfologické analýzy.

Kapitola 2

Syntaktický analyzátor **synt**

V laboratoři, která je součástí Centra zpracování přirozeného jazyka (pracoviště FI MU), je již několik let vyvíjen syntaktický analyzátor **synt** [KH05, Kad07]. Tento analyzátor provádí syntaktickou analýzu typu *chart*, která probíhá na základě příslušné bezkontextové složkové gramatiky¹ a je řízená hlavou pravidel definovaných v této gramatice.

Pro zachování přehlednosti a udržitelnosti je gramatika dvou-, resp. třístupňová (označovaná jako G1, G2, resp. G3), přičemž je využíván fakt, že z jednoho gramatického pravidla lze v některých případech vygenerovat automaticky mnoho dalších na základě předem definovaných metainformací (typu „zpermutuj všechny neterminály na pravé straně pravidla“). Tímto způsobem bylo dosaženo toho, že ručně upravována je pouze metagramatika G1 (obsahující v současné době 239 pravidel), ze které je následně, zpravidla při překladu analyzátoru, vygenerována gramatika G2 (obsahující nyní 3 867 pravidel, jejichž ruční údržba by byla stěží představitelná). Kontextové jazykové prvky (v češtině se například jedná o gramatickou shodu) jsou realizovány prostřednictvím kontextových *akcí* (testů), které jsou přiřazeny k vybraným pravidlům.

Gramatika G3 představuje expandovanou gramatiku G2 v závislosti na příslušných kontextových akcích – všechny akce jsou převedeny do podoby standardních pravidel, tj. z každého pravidla v G2 je vytvořena množina pravidel, jež v sobě přímo nesou informace o pádu, rodu a čísle. Pro běžnou analýzu se však většinou používá gramatika G2 a kontextové akce jsou prováděny nikoli na úrovni gramatiky, nýbrž programově prostřednictvím za tímto účelem vytvořených funkcí a aplikačního rozhraní pro přístup k nim. Tento způsob umožňuje jejich flexibilnější úpravu i kontrolu na úrovni zdrojového kódu bez toho, aby bylo nutné zasahovat do gramatiky – což je velmi užitečné i při vývoji programu, neboť není nutné při úpravách akcí znovu generovat používanou gramatiku.

1. Konkrétně se jedná o gramatiku HPSG – Head-driven phrase structure grammar [Hor01, s. 18 a následující].

Vstupem pro **synt** může být prostý text, vertikální text², text ve formátu **brief**³ nebo zadaný rozsah v některém z dostupných korpusů, které jsou zpřístupněny prostřednictvím korpusového manažeru **manatee** [Ryc00]. V prvních dvou uvedených případech je pro získání základních morfologických informací využit morfologický analyzátor **ajka** [Sed05], v ostatních případech je morfologické značkování součástí vstupu.

Základní výstupem syntaktického analyzátoru **synt** je binární odpověď „akceptuji“/„neakceptuji“, a v kladném případě pak množina platných syntaktických stromů, tj. odvození podle vstupní gramatiky. Přesněji řečeno je výstupem orientovaný graf typu chart (příklad viz obrázek č. A.1 na straně 32), ze kterého je možné syntaktické stromy extrahovat. Jednotlivé hrany tohoto chartu odpovídají použití některého z pravidel ve vstupní gramatice. Pod pojmem chart tedy rozumíme jednak typ datové struktury používané v průběhu analýzy, jednak také výsledek analýzy (tj. stav chartu po analýze).

Současná situace je taková, že **synt** dosáhl při nedávném testování pokrytí převyšujícího 92,08 %, přičemž v přibližně 84 % se mezi výslednými syntaktickými stromy nacházel správný strom (přesně odpovídající odvození dané věty z lingvistického pohledu) [Kad07, s. 77]. Celkový počet syntaktických stromů však bývá i pro krátké věty obrovský – například analýza (poměrně krátké) věty „*Malý hubený chlapec s čepicí jde po ztemnělé mokré ulici pokryté popelem.*“ vrací 486 464 možných stromů odvození, pro větu „*Tyto normy se však odlišují nejen v rámci různých národů a států, ale i v rámci sociálních skupin, proto považují dřívější pojetí za dosti široké a nedostačující.*“ nachází **synt** 1 224 816 možných stromů⁴ a existují věty, výsledkem jejichž analýzy je i více než miliardy výstupních stromů.

Jedná se o daň za dosažení poměrně vysokého pokrytí, které vyžaduje začlenění i pravidel pokrývajících některé velmi specifické okrajové gramatické jevy, přičemž v řadě případů dochází k tomu, že tato pravidla jsou použita i na běžné gramatické struktury, což má za výsledek znásobení počtu výsledných syntaktických stromů (a další nežádoucí vlivy – viz podkapitolu č. 3.3 Faktory ovlivňující zpětnou protragaci).

2. Vertikální text je XML dokument definovaný pomocí DTD na <http://www.fi.muni.cz/nlp/vertical.dtd>, vertikální se nazývá, neboť každé slovo se nachází na samostatném řádku.

3. Formát **brief** se skládá ze tří částí, zpravidla odkazovaných anglicky jako *word-lemma-tag*, tedy slovo, jeho základní tvar a morfologická značka (či obecně značky – může jich být i více v důsledku víceznačného výsledku analýzy).

4. Pro zajímavost: pokud bychom zahrnuli i stromy, které neuspěly při provádění kontextových akcí (tj. zejména kontrol gramatické shody), dostali bychom 1 364 317 344 stromů.

V současné době jsou v **syntu** implementovány dva zásadní mechanismy, jejichž cílem je vypořádat se s problémy, které s sebou velké množství výstupních stromů přináší [KH07, KHKar]:

- **pravděpodobnostní ohodnocení jednotlivých pravidel**, na základě kterého je pak vypočteno i ohodnocení pro výsledný syntaktický strom (podle tohoto ohodnocení jsou na výstupu stromy vypisovány od nejvyššího po nejnižší). Cílem tedy je, aby správný strom získal pokud možno největší ohodnocení a byl vypsán jako první, nebo alespoň mezi několika málo prvními stromy, které je možno ručně zpracovat.
- **diferenciace pravidel do různých úrovní**, přičemž lze zajistit, aby při úspěšné analýze s pravidly vyšší úrovně byly automaticky odstraněny syntaktické stromy obsahující pravidla nižší úrovně. Tímto způsobem lze priorizovat použití některých pravidel a mj. také omezit použití některých velmi specifických pravidel pouze tam, kde jsou skutečně nutná (tj. kde by bez jejich přítomnosti nebyla zadaná věta akceptována).

Kapitola 3

Extrakce syntaktických struktur

3.1 Syntaktické struktury

Jak bylo nastíněno v předchozí kapitole, velký počet syntaktických stromů na výstupu analyzátoru **synt** představuje – zejména praktický – problém, jak takový výstup zpracovat a dále využít. Proto vyvstala potřeba, jejíž naplnění je cílem této práce, totiž využít výsledky syntaktické analýzy k rozšíření výstupu analyzátoru o „strukturní“ pohled na analyzovanou větu. Tento strukturní pohled by měl poskytovat přehled o *větných složkách* [PO92, s. 49] jako jsou jmenné, předložkové či příslovečné fráze.

Abstrahujeme-li od striktně lingvistického pohledu, pak všechny tyto struktury odpovídají na úrovni gramatiky jejím jednotlivým neterminálům. Jejich extrakce pak znamená vhodnou identifikaci hrany (či hran) ve výsledném chartu, jejichž pravidla obsahují na levé straně zadaný neterminál. Z výpočetního hlediska lze chart zpracovávat pro libovolný vstup, byť jsou jeho výstupem leckdy i miliardy syntaktických stromů.

Otázkou zůstává, jakým způsobem tyto struktury ve výsledcích analýzy identifikovat. Vzhledem k tomu, že se tyto struktury vzájemně výrazně odlišují v tom, z jakých částí se skládají (jaká je jejich stavba), není možné použít jednotný způsob pro jejich univerzální extrakci¹. Proto byla extrakce těchto struktur navržena s možností volby několika heuristik (z nichž jedna je výchozí), které umožní uživateli flexibilní určení požadovaného přístupu, vzájemné porovnávání všech dostupných postupů a v budoucnu – podle potřeby – i jejich případné další rozpracování.

Tento přístup má navíc tu výhodu, že umožňuje pracovat s libovolným zadaným neterminálem, není tedy omezen na předem definovanou množinu (která by vyžadovala ruční údržbu v případě změny gramatiky) – jakkoli vzhledem k tomu, že gramatika obsahuje mnoho neterminálů, které plní ryze technickou úlohu (tj. neodpovídají jim přímo žádné jazykové prvky), lze vymezit (z celkového počtu poměrně úzkou) skupinu neterminálů, jejichž

1. Srovnej například stavbu jmenné a slovesné fráze v češtině.

3. EXTRAKCE SYNTAKTICKÝCH STRUKTUR

extrakce přináší z lingvistického pohledu největší přínos, a na tuto skupinu se dále zaměřit při zpracování heuristik a jejich testování. Do této skupiny byly po úvaze zařazeny následující neterminály:

název neterminálu	role v gramatice
np	jmenná fráze
pp	předložková fráze s částicí
pn	předložková fráze
clause	jednoduchá věta
relclause	vztažná věta
abbr	zkratka
inter	intersegment ²
np_prop_names	vlastní jméno
numk	číslovka
part_adv	přísluvečná skupina s částicí
vp ³	slovesná fráze

Tabulka 3.1: Seznam neterminálů významných z hlediska extrakce syntaktických struktur.

Není-li stanoveno jinak, pak výchozí postup spočívá v nalezení *největších* neterminálových struktur (největších vzhledem k inkluzi, uvažujeme-li větu jako množinu slov). Toto východisko nalézá dobré uplatnění v syntaktických strukturách jako jsou jmenné či předložkové fráze. Výsledkem je komplexní pohled na danou větu z hlediska výskytu hledané struktury. Tento postup ale není vždy žádoucí – zejména v důsledku silně nejednoznačné morfologické analýzy může vést k extrakci velkých (chybných) struktur, které nemají velkou vypovídací hodnotu.

Duálně k výše uvedenému je implementován také postup opačný, tj. extrakce *nejmenších* struktur (opět vzhledem k inkluzi, jak je uvedeno výše). Tento přístup se může hodit, chceme-li získat holý větný člen nějaké syntaktické struktury. Je tomu tak proto, že mnoho z námi studovaných neterminálů (tj. syntaktických struktur) je (zleva, zprava, případně z obou stran) přímo rekurzivních, tj. gramatika obsahuje pro tyto neterminály pravidla tvaru $A \rightarrow A\alpha \mid \varepsilon$ (v případě levé rekurze). Holý větný člen takové struktury je pak podstruktura identifikovaná hranou, která vznikla uplatněním pravidla $A \rightarrow \varepsilon$.

2. Cokoliv, co může být vloženo mezi členy slovesné fráze.

3. Tento neterminál je přidán jako svým způsobem „virtuální“ a představuje v podstatě heuristiku pro zpracování slovesných frází. V gramatice se ve skutečnosti nevyskytuje, protože se ovšem jedná o podstatnou lingvistickou strukturu, rozhodl jsem se pro toto řešení.

3. EXTRAKCE SYNTAKTICKÝCH STRUKTUR

Pro ilustraci uvádím níže v tabulce č. 3.2 srovnání obou přístupů na větě „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.“ pro hledaný neterminál **np** (jmenná fráze):

č. ⁴	struktura podle „největší“
[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
č.	struktura podle „nejmenší“
[0-1)	Tyto
[1-2)	normy
[2-3)	se
[7-8)	rámci
[8-9)	různých
[9-10)	národů
[11-12)	států
[16-17)	rámci
[17-18)	sociálních
[18-19)	skupin
[23-24)	dřívější
[24-25)	pojetí
[27-28)	široké
[29-30)	nedostačující

Tabulka 3.2: Porovnání přístupů hledání největších a nejmenších struktur.

Tento příklad dobře ilustruje, že právě pro jmenné fráze dává postup „největší“ při jednoznačné (a samozřejmě správné) morfologické analýze optimální výsledek. I při nejednoznačné analýze je jeho výstup vhodnější pro další zpracování, a proto se jeví jako výhodnější výchozí volba.

Výjimku tvoří neterminály **relclause** (vztažná věta) a **abbr** (zkratka), pro které je jako výchozí používán postup „nejmenší“. V prvním zmíněném případě je tomu tak proto, že zpravidla chceme identifikovat jednotlivé vztažné věty (i zanořené), ve druhém případě je důvod zřejmý – zkratky představují ze své podstaty izolované útvary, jejichž skupiny nenesou žádný význam.

4. Číslování vymezuje (zleva uzavřený, zprava otevřený) interval dané struktury v rámci vstupní věty (jedná se tedy o indexy slov).

Kromě těchto dvou základních heuristik je dále uplatňováno několik postupů specifických pro určité neterminály, případně dostupných jako volby při spuštění. Tyto dílčí techniky se snaží optimalizovat hledání určitých specifických (pod)struktur identifikovaných následujícími neterminály:

- neterminál **vp** (slovesná fráze⁵)

Za účelem extrakce slovesných frází jsou nejprve vytvořeny struktury neterminálů **clause** a **relclause** (podle heuristiky „nejmenší“), ve kterých jsou následně identifikovány slovesné preterminály⁶ a odpovídající terminální řetězce (tj. slova), která pak (ve stejném pořadí jako ve větě) tvoří slovesnou frázi v rámci podstruktur **clause** a **relclause**. Výstupem pro větu „*Kdybych to byl býval věděl, byl bych sem nechodil.*“ jsou slovesné fráze:

[0-5): byl býval věděl

[6-10): byl bych nechodil

- neterminál **clause** (jednoduchá věta)

V případě tohoto neterminálu je uplatněna jako výchozí heuristika „nejmenší“, přičemž jsou procházeny jak struktury neterminálu **clause**, tak i **relclause** (vztažné věty). Výsledek je získán tak, že z jsou nalezených struktur vyjmuty jejich podstruktury. Tímto způsobem lze spolehlivě identifikovat jednoduché věty pro souvětí typu „*Vidím ženu, která drží růži, která je červená.*“:

[0-3): Vidím ženu ,

[3-7): která drží růži ,

[7-10): která je červená

Týž postup lze použít i pro typ „*Muž, který stojí u cesty, vede kolo.*“:

[0-9): Muž , , vede kolo

[2-6): který stojí u cesty

- libovolný neterminál společně s volbou **-m** (morfologické členění)

V tomto případě se jedná o univerzální volbu pro libovolný neterminál, která umožňuje následné rozdělení (pod)struktur podle výsledků morfologické analýzy. Za tímto účelem byly výsledky syntaktické analýzy využity pro zpřesnění analýzy morfologické, jak je blíže vysvětleno v následující kapitole. Jednotlivé syntaktické struktury jsou v tomto

5. Viz poznámku 3 pod tabulkou č. 3.1 na straně 11.

6. Preterminál je neterminál vyskytující se na levé straně pravidla tvaru $V \rightarrow \alpha$, kde α je terminální řetězec.

případě rozděleny tak, aby v rámci struktury platila gramatická shoda v pádě, čísle a rodě. Jako příklad uvádím výstup pro větu „*Představa takového postupu bez potřebného technického zajištění je nemyslitelná.*“:

[0-1): Představa

[1-3): takového postupu

[3-7): bez potřebného technického zajištění

Již dříve zmíněná věta „*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.*“ je pomocí této volby rozčleněna následovně:

[0-4): Tyto normy se však

[6-8): v rámci

[8-12): různých národů a států

[12-13): ,

[13-17): ale i v rámci

[17-19): sociálních skupin

[19-20): ,

[23-25): dřívější pojetí

[25-30): za dosti široké a nedostačující

3.2 Zpětné zpřesnění výsledků morfologické analýzy

Poslední z možných postupů popsanych v předchozí podkapitole, jak identifikovat hledané syntaktické (pod)struktury, morfologické členění, využíval morfologické analýzy jednotlivých slov. Ta je sice – nejsou-li vstupní data přímo jednoznačně označovaná – silně nejednoznačná, jak jsem nastínil v části věnované této problematice na straně č. 4, k jejímu zpřesnění však lze použít nové informace, které s sebou přináší syntaktická analýza (jakkoli je také nejednoznačná).

Jde především o znalost gramatických shod v rámci některých slovních frází – může-li adjektivum „*zvláštní*“ představovat z morfologického hlediska 27 různých tvarů, pak ve slovním spojení „*zvláštní ukaz*“ připadají v úvahu už pouze nominativ a akuzativ singuláru maskulina a ve spojení „*na zvláštní ukaz*“ již lze spolehlivě identifikovat akuzativ (analýza je tedy pro všechna tři slova jednoznačná!).

Z pohledu syntaktické analýzy se fakticky jedná o *zpětnou propagaci* morfologických hodnot směrem od vrcholu chartu k jeho terminálním uzlům (tj. slovům). V průběhu analýzy jsou totiž při vyhodnocování kontextových akcí tyto informace propagovány směrem opačným a jsou využívány k eliminaci

odvození, která kontextové podmínky (tj. zejména gramatické shody) nesplňují (hovoříme o tom, že kontextové akce „prořezávají“ množinu výsledných syntaktických stromů). Dosud však takto získané informace nebyly využívány ke zpětnému zpřesnění morfologické analýzy jednotlivých slov.

Jak ukázalo testování na necelých 30 000 větách z korpusu DESAM [PRS97], v průměru se po provedení tohoto zpětného zpřesnění vyskytuje ve větě o 30 % více morfologicky jednoznačně analyzovaných slov než před ním:

ukazatel	před	po
procentuální průměr jednoznačně analyzovaných slov	20,733 %	46,1172 %
průměrně odstraněno značek ve větě	38,3716 %	
chybovost ⁷	< 1,46 %	
počet vět	29 604	

Vzhledem k nízké chybovosti se ukazuje, že se jedná pravděpodobně o jednu z nej přesnějších metod částečné morfologické desambiguace, která navíc nespočívá v přidání žádných explicitních pravidel, nýbrž vychází přímo z použité gramatiky. Běžné techniky částečné/úplné desambiguace jsou založeny na kombinaci statistických a pravidlových prvků (tato pravidla mohou být ručně vytvořena – viz např. [HKK⁺01], případně automaticky generovaná na základě kontextu daného slova v korpusu [Šm04]), tedy v podstatě heuristik, byť velmi sofistikovaných a úspěšných.

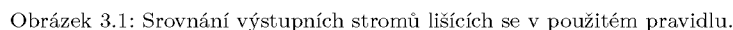
Navržený způsob samozřejmě nelze použít pro úplnou desambiguaci a zdaleka nedosahuje takového pokrytí, jako výše zmiňované metody, má ovšem jednu podstatnou výhodu: nejedná se o heuristiku a z teoretického pohledu by tento způsob za předpokladu správné a přesné gramatiky měl dosahovat u jazykových jevů, které se snaží pokrýt, 100% úspěšnosti.

Skutečnost, že se v provedených testech úspěšnost skutečně blíží 100 % ukazuje tedy mimo jiné na fakt, že gramatika **syntu** a zejména přiřazení jednotlivých kontextových akcí je velmi přesné.

7. Za chybu je považováno, pokud během zpětné propagace dojde k odstranění správné značky (korpus DESAM je označovaný a poskytoval tedy možnost srovnání). Ve skutečnosti je chybovost ještě menší, neboť za chybné byla řada výsledků označena i v důsledku chyby ve značce v korpusu.

3.3.1 Rozlišovací schopnost gramatiky

Pro příklad níže uvádím první dva⁸ výstupní stromy pro vstupní větu „*Na krásné dlouhé ulici stálo malé modré auto.*“:



8. Podle pravděpodobnostního ohodnocení stromů na výstupu analyzátoru – viz kapitola č. 2.

3. EXTRAKCE SYNTAKTICKÝCH STRUKTUR

skutečnosti má toto pravidlo pokrývat zcela jiný gramatický jev a nemělo by tedy být v tomto případě použito⁹. Výsledek na větě „*Na krásné dlouhé ulici stálo moderní nablýskané auto.*“, kde k podobnému jevu nedošlo, je vidět v následující tabulce:¹⁰

slovo	před	po
Na	k7{c4, c6}	k7c6
krásné	k2eA{gFnPc1d1, gFnPc4d1, gFnPc5d1, gFnSc2d1, gFnSc3d1, gFnSc6d1, gInPc1d1, gInPc4d1, gInPc5d1, gInSc1d1wH, gInSc4d1wH, gInSc5d1wH, gMnSc4d1, gMnSc1d1wH, gMnSc5d1wH, gNnSc1d1, gNnSc4d1, gNnSc5d1}	k2eAgFnSc6d1
dlouhé	k2eA{gFnPc1d1, gFnPc4d1, gFnPc5d1, gFnSc2d1, gFnSc3d1, gFnSc6d1, gInPc1d1, gInPc4d1, gInPc5d1, gInSc1d1wH, gInSc4d1wH, gInSc5d1wH, gMnSc4d1, gMnSc1d1wH, gMnSc5d1wH, gNnSc1d1, gNnSc4d1, gNnSc5d1}	k2eAgFnSc6d1
ulici	k1gFnSc3, k1gFnSc4, k1gFnSc6	k1gFnSc6
stálo	k5eAaImAgNnSaIrD	k5eApNnStMmPaI ¹¹
moderní	k2eA{gFnPc1d1, gFnPc4d1, gFnPc5d1, gFnSc1d1, gFnSc2d1, gFnSc3d1, gFnSc4d1, gFnSc5d1, gFnSc6d1, gFnSc7d1, gInPc1d1, gInPc4d1, gInPc5d1, gInSc1d1, gInSc4d1, gInSc5d1, gMnSc1d1, gMnSc4d1, gMnSc5d1, gMnSc1d1, gMnSc5d1, gNnSc1d1, gNnSc4d1, gNnSc5d1}	k2eAgNnSc1d1, k2eAgNnSc4d1, k2eAgNnSc5d1
nablýskané	k2eA{gFnPc1d1rD, gFnPc4d1rD, gFnPc5d1rD, gFnSc2d1rD, gFnSc3d1rD, gFnSc6d1rD, gInPc1d1rD, gInPc4d1rD, gInPc5d1rD, gInSc1d1wHrD, gInSc4d1wHrD, gInSc5d1wHrD, gMnSc4d1rD, gMnSc1d1wHrD, gMnSc5d1wHrD, gNnSc1d1rD, gNnSc4d1rD, gNnSc5d1rD}	k2eAgNnSc1d1, k2eAgNnSc4d1, k2eAgNnSc5d1
auto	k1gNnSc1, k1gNnSc4, k1gNnSc5	k1gNnSc1, k1gNnSc4, k1gNnSc5

Tabulka 3.3: Srovnání morfologického ohodnocení před a po zpětném zpřesnění.

Jak je z tabulky patrné, morfologické zpřesnění je v tomto případě skutečně výrazné: prvních pět slov je určeno zcela jednoznačně, zbylá tři pak jednoznačně určená nejsou, přesto ale i u nich je posun výrazný – k jejich jednoznačnému určení by v tomto případě vedlo např. ověření valenčního rámce slovesa „*stát*“ – tato funkcionalita ovšem zatím není implementována.

9. Detailní popis pravidel gramatiky syntu by značně přesahoval rámec této práce.

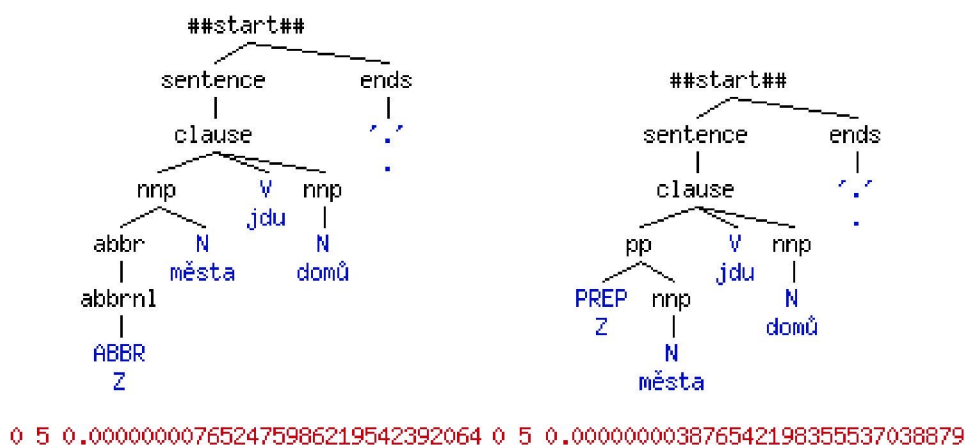
10. Pro stručnost je zavedena notace $k1\{gFnPc1, gMnSc1\} \cong k1gFnPc1, k1gMnSc1$.

11. Nesrovnalost ve značkách na tomto řádku má ryze technický důvod (zastaralá knihovna pro jejich zpracování) a není pro ilustrovaný účel podstatná.

3.3.2 Morfologické marginálie

Druhým podstatným faktorem, který dokáže významně ztížit jak vlastní syntaktickou analýzu, tak sekundárně i zpětné zpřesnění morfologických výsledků, jsou *morfologické marginálie*, tedy jazykově zcela okrajové morfologické jevy, které ovšem, pokud jsou morfologickým analyzátozem brány v potaz, dokáží zásadním způsobem znehodnocovat výsledky. Toto postihuje i morfologický analyzátor (**ajka**) používaný **syntem** – v současné době zahrnuje ve své databázi velmi různorodé případy toho, co jsem si dovolil nazvat marginálií, ve snaze poskytnout co možná nejúplnější morfologickou analýzu postihující morfologii češtiny ve vší rozmanitosti. Například pro vstupní slovo „z“ nachází dva možné slovní druhy: předložku a citoslovce.

Současný stav je takový, že oba možné tvary vstupují jako zcela rovnocenné do procesu syntaktické analýzy, přestože druhý z nich se zřejmě vyskytuje naprosto minimálně. Důsledkem tohoto postupu je opět to, že např. předložková skupina spjatá s předložkou „z“ nebude prořezána na jediný možný pád, který tato předložka dovoluje (tj. genitiv), protože mezi platnými odvozeními bude s velkou pravděpodobností i strom uvažující „z“ jako citoslovce, jak pro názornost ukazují stromy na vstupní větě „Z města jdu domů.“:



Obrázek 3.2: Srovnání výstupních stromů lišících se v použitém pravidlu.

V tomto případě byl dokonce špatný strom (vlevo) vyhodnocen jako pravděpodobnější a z celkového počtu 8 výstupních stromů obsahuje 6 z nich špatně analyzované první slovo.

Takových a podobných případů je bohužel velmi mnoho – všímavý čtenář si jistě v předchozím textu pojednávajícím o nepřesnosti gramatiky všiml, že ve jmenné skupině „*malé modré auto*“ byla pro první dvě slova ohodnocena jednak jako adjektiva, jednak jako substantiva¹². I v tomto případě se jedná o dosti marginální jazykový jev (byť ne tak výrazně, jako v předchozím případě).

Nabízí se jistě několik možných řešení všech těchto „marginálních“ problémů, zde uvádím alespoň dva základní, jež se zásadně liší ve své koncepci:

- **redukce na základě četnosti výskytů v korpusu**
Poměrně přímočarý způsob spočívá ve využití dostupných informací z označovaného korpusu a zahození některých výsledků morfologické analýzy, pro které je četnost výskytu daného slova s nalezenou značkou (potažmo pouze slovním druhem) v korpusu pod určitým pevně stanoveným limitem. Toto řešení má však jednu poměrně zásadní nevýhodu, vyžaduje totiž přístup ke korpusu během analýzy (což značně omezuje využití syntaktického analyzátoru), případně přístup k databázi četností předem vyextrahovaných z korpusu (jež by byla poměrně objemná a musela by být součástí analyzátoru).
- **redukce na základě systému pravidel**
Zcela jiný přístup představuje postup spočívající ve vytvoření sady pravidel určujících přednosti morfologických ohodnocení (či spíše jenom slovních druhů) bez ohledu na konkrétní slova – např. „existuje-li ohodnocení jako substantivum i jako adjektivum, vyber pouze adjektivum“, nebo „existuje-li ohodnocení jako citoslovce a libovolný jiný slovní druh, vyber pouze tento jiný slovní druh“.

Ať už bude použit pro redukcí marginálií jakýkoliv postup, vždy bude ovšem potřeba vytvořit účinný mechanismus, jak zabránit snížení pokrytí syntaktického analyzátoru v důsledku této redukce – účinný, ovšem časově náročný přístup, představuje realizace syntaktické analýzy dvoufázově: poprvé s redukcí marginálií a v případě neakceptování (nenalezení žádného odvozovacího stromu) znovu se všemi morfologickými výsledky.

12. Tedy „*to malé*“, „*to modré*“.

Kapitola 4

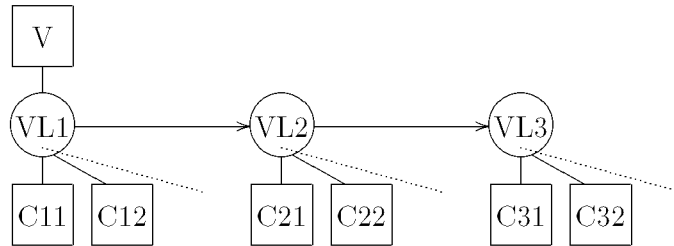
Implementace

Implementace výše popisovaných postupů v programovacím jazyku C se (jako součást syntaktického analyzátoru **synt**) nachází v souboru **extras.c** s příslušnými deklaracemi v **extras.h**, jakkoli bylo nutné provést řadu dílčích úprav na mnoha jiných místech zdrojového kódu **syntu**. Úplný výčet všech funkcí (tj. obsah souboru **extras.h**) se nachází v dodatku B na straně 33. Pracovní tok programu sestává především z průchodu výsledného chartu po předchozí syntaktické analýze, během kterého je (v rámci jediného průchodu) provedena zpětná propagace výsledků morfologické analýzy a extrakce struktur pro zadaný neterminál. V následujících podkapitolách proto uvádím bližší popis těchto tří – více či méně samostatných – částí.

4.1 Průchod chartem

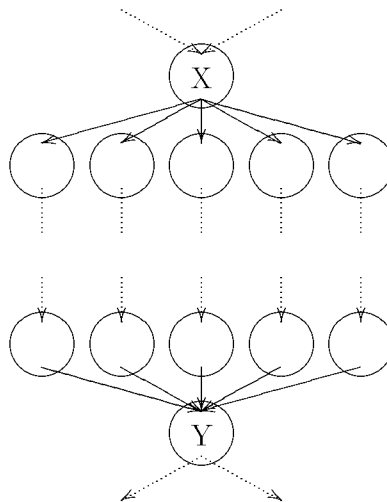
Zásadní charakteristikou chartu je jeho schopnost komprimace obrovského množství syntaktických stromů do řádově menší struktury, což umožňuje efektivně provést jak vlastní syntaktickou analýzu, tak i další zpracování jejího výstupu. Intuitivně si lze chart představit jako les stromů, ve kterém jsou hrany a uzly společné více stromům ztotožněny a vyskytují se v něm tedy pouze jednou (společné však mohou být pro libovolně mnoho stromů). Zatímco počet syntaktických stromů může být až exponenciální vzhledem k počtu slov ve větě, struktura chartu obsahuje vždy polynomiálně mnoho hran [Hor01, s. 133] a její zpracování tedy nepředstavuje z hlediska výpočetní složitosti problém.

Pro zpracování výsledného chartu není přímo použita stejná datová struktura, která je využívána pro algoritmus vlastní syntaktické analýzy, nýbrž tzv. les hodnot (*forest of values* – viz obrázek 4.1), jehož použití pro další zpracování je mnohem přímočařejší (pro srovnání a popis viz [KHKar, s. 3–5]). V termínu les hodnot znamenají hodnoty výsledky morfologické analýzy – každá tato hodnota odpovídá právě jednomu uzlu ve výsledném chartu, obě struktury jsou tedy prakticky paralelní, pro procházení směrem shora dolů je ovšem les hodnot mnohem vhodnější.



Obrázek 4.1: Struktura les hodnot (*forest of values*) – každá hodnota (V) odkazuje na pole seznamů (VL) dceřiných hodnot (C).

Pro průchod výše ilustrovaného lesu hodnot – orientovaného acyklického grafu – je možné použít obě základní techniky prohledávání do hloubky i do šířky. První možnost je ovšem z hlediska časové složitosti značně neoptimální, protože v grafu se mohou vyskytovat (a v praxi se velmi často vyskytují!) podgrafy, jež se větví z jednoho společného vrcholu a tyto větve se později opět spojí v nějakém společném vrcholu (viz obrázek 4.2). Při prohledávání do hloubky jsou pak všechny uzly pod tímto společným vrcholem (na obrázku označen jako Y) procházeny (zbytečně) vícekrát – tolikrát, kolik vede do tohoto vrcholu hran (stojí za zdůraznění, že vznik takové situace nezávisí na počtu výstupních stromů, nýbrž na jejich vnitřní topologii, která je dána gramatikou a nelze ji ovlivnit způsobem výpočtu).



Obrázek 4.2: Ilustrace podgrafu problematického z hlediska použití procházení do hloubky.

Není těžké najít příklad, při kterém má tento jev za následek řádový nárůst časové složitosti, z časového hlediska se proto jako mnohem výhodnější jeví pro tento účel procházení do šířky, byť může být vykoupeno většími paměťovými nároky, neboť v určitých situacích je při procházení do šířky třeba uchovat v paměti všechny uzly grafu.

Vzhledem k tomu, že je ovšem využita stávající struktura (les hodnot), která už se stejně celá (tj. všechny uzly) nachází v paměti po předchozí syntaktické analýze, není nárůst použité paměti nikterak veliký – v podstatě se v paměti nachází navíc akorát ukazatele mezi jednotlivými hodnotami, jež vytvářejí spojový seznam používaný jako fronta typu FIFO¹ (s kontrolou zamezující opakovanému vkládání téže hodnoty). Na této frontě je následně standardním způsobem realizováno procházení do šířky.

Srovnání obou přístupů na větě „*Etopedie, jakožto speciálně-pedagogická disciplína se zabývá edukací, reedukací a prevencí u jedinců s rizikem poruch chování a emocí i u jedinců, kteří již těmito poruchami trpí.*“ ukazuje následující tabulka (v tomto případě je rozdíl v době výpočtu skutečně významný, aniž by to ovlivnilo paměťové nároky – jak ukazuje obrázek č. 4.3, maximum alokace, tedy tzv. *memory peak*, nastává totiž dříve, už během procesu vlastní syntaktické analýzy):

ukazatel	do hloubky	do šířky
čas ²	23 min 2,156 s	0,837 s
paměť ³	51,05 MB	51,05 MB

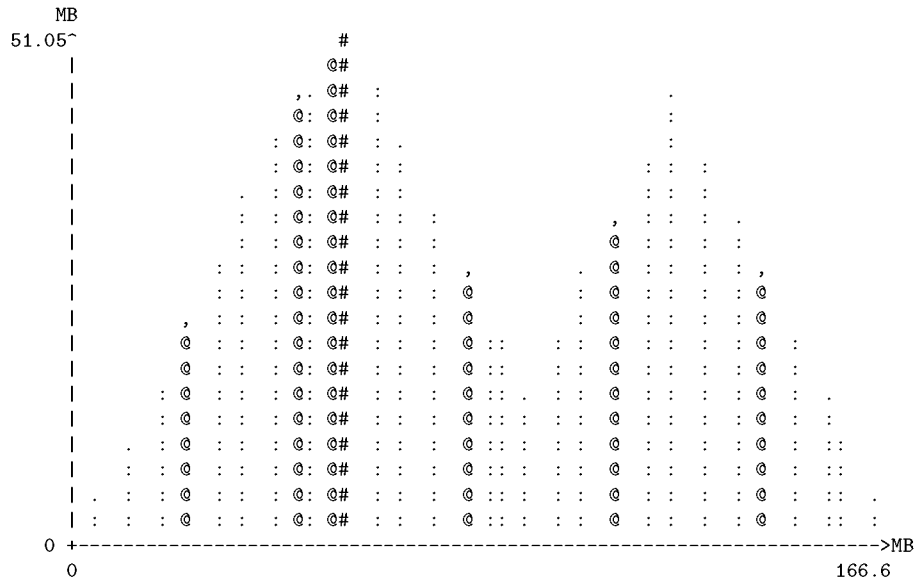
Tabulka 4.1: Srovnání prohledávání do hloubky a do šířky na vybrané větě.

Výchozím přístupem je tedy procházení do šířky, přičemž použitím makra `PROPAGATE_DFS` při kompilaci `syntu` lze vynutit procházení do hloubky. V praxi ale nepředpokládám jeho využití pro jiné než testovací účely – snad s výjimkou případu, kdy by bylo nutné použít procházení do hloubky v extrémních případech vyžadujících práci s minimem operační paměti. Smysl to bude mít samozřejmě pouze za předpokladu, že maximum alokované paměti nenastane už během předchozí syntaktické analýzy, jako je tomu v tomto případě (a jiných, které jsem testoval).

1. *First in, first out* – blíže např. <http://en.wikipedia.org/wiki/FIFO>.

3. Pro měření času byl použit standardní unixový program `time`, test probíhal na stroji s 4 GB RAM a CPU Intel Xeon 2 GHz.

3. Jedná se o maximum paměti alokované na haldě v průběhu vykonávání programu a pro jeho měření byl použit program `valgrind` (konkrétně jeho modul `massif`) – dostupný z <http://www.valgrind.org>.



Obrázek 4.3: Množství alokované paměti na haldě v závislosti na postupném průběhu alokace (výstup pochází opět z programu `valgrind`).

Jak je patrné z obrázku č. 4.3, v průběhu chodu programu jsou zřetelné dva alokační vrcholy. K prvnímu dochází v průběhu syntaktické analýzy, ke druhému pak v důsledku dočasné duplikace morfologického ohodnocení u všech uzlů v lese hodnot (viz následující podkapitolu).

4.2 Zpětná propagace výsledků morfologické analýzy

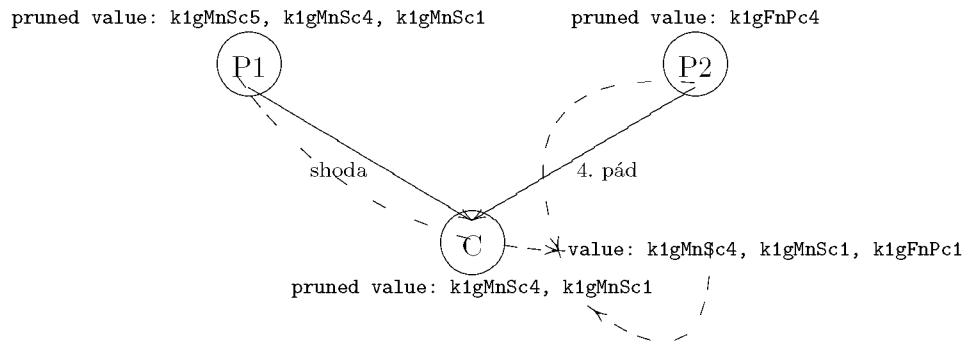
Zpětná propagace výsledků morfologické analýzy probíhá při průchodu lesem hodnot na základě kontextových akcí definovaných v gramatice (viz kapitulu 2 na straně 7) – v podstatě tedy zrcadlí akce, které probíhají v průběhu analýzy, ovšem v opačném směru (shora dolů). V úvahu nejsou brány všechny akce, nýbrž jenom ty, které mohou nějakým způsobem ovlivnit morfologické ohodnocení (konkrétně se jedná o testy na určitý pád, či na úplnou nebo částečnou gramatickou shodu v pádě, rodě a čísle).

Zpětnou propagaci obstarává v rámci nelistových uzlů lesu hodnot funkce `propagate_values_to_leaves_bfs()`⁴ a z listů lesu hodnot do seznamu slov (který je reprezentovaný strukturou tvar-lemma-značky) pak funkce `prune_word_values()`.

4. Případně `propagate_values_to_leaves_dfs()` v případě volby procházení do hloubky pomocí kompilačního makra `PROPAGATE_DFS`, viz podkapitolu 4.1.

V průběhu zpětné propagace (ať už je les hodnot procházen do hloubky či do šířky) je třeba zajistit korektní propagaci všech možných hodnot: vzhledem k topologii chartu jsou totiž uzly procházeny zpravidla vícekrát – tolikrát, kolik do nich vede hran. Proto je v každém uzlu nejdříve vyvořena pomocná hodnota (**pruned value**), která je prázdná a je postupně naplňována v průběhu zpětné propagace. Vždy jsou nejprve původní hodnoty (**value**) v daném uzlu prořezány podle nadřazeného uzlu (v závislosti na kontextové akci, která je s příslušnou hranou spojena) a výsledek je sloučen do pomocné hodnoty, ze které je pak propagován do nižších pater grafu až k listům.

Průběh zpětné propagace ze dvou rodičovských uzlů ilustruje následující obrázek:



Obrázek 4.4: Průběh zpětné propagace morfologických hodnot.

S hranou vedoucí z rodičovského uzlu P1 do potomka C je asociována kontextová akce vyžadující úplnou gramatickou shodu v pádě, rodě a čísle. V důsledku zpětné propagace dojde u potomka C k odstranění značky **k1gFnPc1**, která se neshoduje v rodě. Zbylé dvě značky jsou uloženy do **pruned value**. Obdobně je s hranou vedoucí z P2 do C asociována kontextová akce spočívající v testu na 4. pád – to vede k odstranění nominativních značek **k1gMnSc1**, **k1gFnPc1** a ke sloučení zbylé značky do **pruned value** (kde už je stejně uložena, fakticky se tedy její obsah nezmění). Další propagace z uzlu C na jeho následníky pak samozřejmě probíhá ze značek uložených v **pruned value**.

Průchod chartem, provedení zpětné propagace morfologických hodnot a vlastní extrakce struktur lze psuedokódem znázornit následovně (varianta s použitím procházení do šířky):

Funkce MAIN

```
1 begin
2   value = start; // Inicializace na vrchol lesu hodnot
3   while value ≠ NULL do
4     PROPAGATE(value); // Propaguj z této hodnoty na její
      potomky
5     value = QUEUE_POP(); // Odeber další hodnotu z fronty
6   end
7 end
```

Funkce PROPAGATE

```
1 begin
2   EXTRAS(value); // Proveď extrakci struktur
3   foreach valueList ∈ value do
4     foreach childValue ∈ valueList do
5       PROPVALUE(value, childValue); // Propaguj morfologické
        ohodnocení podle akcí z value do childValue
6       if NOT_ENQUEUED(childValue) then
7         QUEUE_PUSH(childValue); // Vlož potomka do fronty ke
          zpracování
8       end
9     end
10  end
11 end
```

4.3 Extrakce struktur

4.3.1 Algoritmy

Extrakce neterminálové struktury probíhá souběžně se zpětnou propagací morfologických hodnot, resp. ve chvíli, kdy v daném uzlu již byla provedena zpětná propagace hodnot (a případně tedy prořezány některé morfologické značky). Je zavolána funkce `extras_edge()`, která podle hledaného neterminálu (parametr `-n`), případně dalších volitelných parametrů (`-m`, `-s`) rozhodne, zda je třeba předaný uzel (jednoznačně identifikující určitou strukturu) zpracovat či nikoli a pokud ano, jakým způsobem. V mnoha případech je totiž nutné zpracovávat i jiné neterminály, než jen ten, jehož strukturu hledáme (viz podkapitolu 3.1 na straně 10).

Určit, jakým způsobem má být uzel extrahován, znamená ve většině případů předat jeho zpracování buď funkci `extras_large()` (výchozí volba) nebo `extras_small()` podle toho, zda je cílem nalézt největší/nejmenší (pod)struktury. Obě tyto funkce procházejí doposud identifikované struktury a upraví je podle nově nalezené za předpokladu, že je to vhodné – tedy jestliže nově vkládaná je větší, resp. menší než předcházející. Nejedná se přitom o tak přímočarý postup, jak to na první pohled může vypadat: musíme vzít v potaz, že struktury nejsou zpracovávány v žádném určitém pořadí (během zpracovávání nejsou seřazeny vzhledem ke svému počátku nebo konci), ale tak, jak je procházen les hodnot.

V prvním případě (hledáme největší struktury) je postup ještě poměrně jasný: zpracováváme-li strukturu vymezenou indexy `[left, right)`, pak stačí zkontrolovat, zda neexistuje struktura začínající na pozici $i \leq \text{left}$ a končící na pozici $j \geq \text{right}$. Pokud ne, pak `[left, right)` je největší dosud nalezená struktura.

V případě hledání nejmenších struktur je situace obdobná, přičemž se nabízí řada možností, jak řešit např. kolize vznikající při vzájemně se částečně překrývajících strukturách (máme-li struktury `[left, right)` a `[left+5, right+5)`, jaký je vhodný postup?). V současné době je postup striktně dán tak, že struktura musí být každopádně výsledkem předchozí syntaktické analýzy, nemůže být dodatečně vypočtena, byť v dobré víře ve zlepšení výsledku. Důvody jsou praktické (nečinit celý postup více komplikovaný, než je nezbytně nutné) i principiální: cílem není zlepšovat syntaktickou analýzu, nýbrž v maximální míře vytěžit a zpracovat její stávající výsledky – případné zjištěné nedostatky naopak poslouží k ladění procesu syntaktické analýzy, případné úpravě gramatiky apod.

Po projití celého výstupního chartu (resp. lesu hodnot) a extrakci příslušných struktur je na závěr, je-li zadána volba **-m**, provedeno rozdělení struktur podle gramatické shody. Probíhá velmi jednoduše: každá výsledná struktura je procházena a ve chvíli, kdy se mezi některými jejími dvěma slovy nenajde shoda⁵, je v tomto místě (tj. mezi těmito slovy) daná struktura rozdělena ve dvě samostatné.

4.3.2 Datové struktury

Po zralé úvaze, jak reprezentovat jednotlivé syntaktické struktury, jsem se rozhodl pro minimalistický přístup, který umožní – v duchu extrémního programování – transparentní implementaci a v případě potřeby flexibilní nahrazení vhodnější alternativou. Proto jsou struktury reprezentovány jako prosté pole celých čísel, přičemž index označuje začátek struktury, hodnota na daném indexu pak její konec. Taková pole jsou v průběhu využívána dvě, jedno hlavní a jedno pomocné.

Výše uvedená reprezentace neumožňuje uchovat více struktur začínajících na stejném místě, hodí se proto pro uchovávání výsledných výstupních struktur, které už byly zpracovány. Za účelem uchování všech dostupných informací o strukturách je použito třetí, dvourozměrné pole, u kterého první index označuje začátek, druhý konec a hodnotou je 1 resp. 0 podle toho, zda se v daném rozsahu v chartu vyskytuje resp. nevyskytuje hledaná (pod)struktura. Těto struktury je využíváno mj. proto, aby se při „výpočtu“ struktur (tj. dělení, slučování apod.) zamezilo vytvoření struktury, která se v chartu nevyskytuje.

5. Tj. neexistuje taková kombinace morfologických značek u obou slov, která by se shodovala v pádě, čísle a rodě.

Kapitola 5

Další vývoj

Využití extrakce struktur může být poměrně různorodé, a proto vidím další vývoj především v přizpůsobování a rozšiřování možností extrakce specifickým dílčích úkolů, pro které bude využívána. Tento fakt jsem se samozřejmě snažil zohlednit i v implementaci, jejíž budoucí úpravy by měly vyžadovat pouze dopisování pravidel pro extrakci ve funkci `extras_edge()` a případně modifikaci funkce `print_structures()` zajišťující výpis struktur.

Z hlediska dalšího vývoje syntaktického analyzátoru **synt** vidím jako podstatné další zpřesňování syntaktické analýzy (tj. redukování počtu výstupních stromů). Za tímto účelem se jeví jako velmi žádoucí zahrnout do procesu analýzy podporu pro valence sloves ze vznikajícího slovníku **Verbalex** [HHK06] a dále se zaměřit na optimalizaci používané gramatiky – zejména s ohledem na nastavení úrovní jednotlivých pravidel. Právě větší diferenciací pravidel do jednotlivých úrovní (tedy zejména odsunutí některých pravidel velmi specifických pro poměrně řídké jazykové jevy do vyšších úrovní) s již implementovaným prořezáváním výstupních stromů podle úrovní pravidel (viz kapitolu č. 2) by mohla značně snížit počet výstupních syntaktických stromů.

Pro další efektivní vývoj **syntu** je ovšem třeba vyvinout kvalitní testovací platformu na poloautomatické bázi, pomocí níž bude možné okamžitě vyhodnocovat dopady jakýchkoli úprav gramatiky, aby se zamezilo regresím v analýze.

Vzhledem k tomu, že syntaktický analyzátor **synt** je vyvíjen již bezmála osm let a za tu dobu se na jeho vývoji podílela řada autorů, vidím jako podstatný úkol do budoucna i provedení řady servisních prací ve zdrojovém kódu s cílem zajistit jeho stabilitu a udržitelnost do dalších let.

Závěr

Výsledkem této práce je rozšíření (nejednoznačné) syntaktické analýzy o další typ výstupu, jenž se snaží poskytnout (v maximální možné míře jednoznačný) strukturní pohled na výstavbu věty. Tento výstup může být jednak z uživatelského pohledu leckdy přístupnější než množina syntaktických odvození, jednak se nabízí jako východisko pro další zpracování za účelem extrakce informací. Možné je i jeho využití coby užitečného nástroje pro různé dílčí úkoly v rozmanitých oblastech zpracování přirozeného jazyka (aktuálně se jedná např. o převod vět do struktury **predikát(argumenty)** pro další logickou analýzu). Využívaná datová struktura – chart – je masivně nejednoznačná, vzhledem k počtu slov ve vstupní větě ovšem pouze polynomiálně velká, a proto je vytvořená implementace z hlediska výpočetní složitosti efektivní a lze ji použít pro zpracovávání velkého objemu dat.

Dalším významným výsledkem je i implementace zpětného zpřesnění morfologické analýzy a vyhodnocení přínosu této techniky, kterou lze použít např. pro přesnější automatické značkování korpusu, nebo alespoň pro zpřesnění značkování před jeho ručním provedením lingvistou. Zhotovená měření ukazují, že se jedná o vysoce spolehlivý postup, jak docílit (správného) jednoznačného morfologického ohodnocení v průměru u téměř poloviny slov ve vstupní větě.

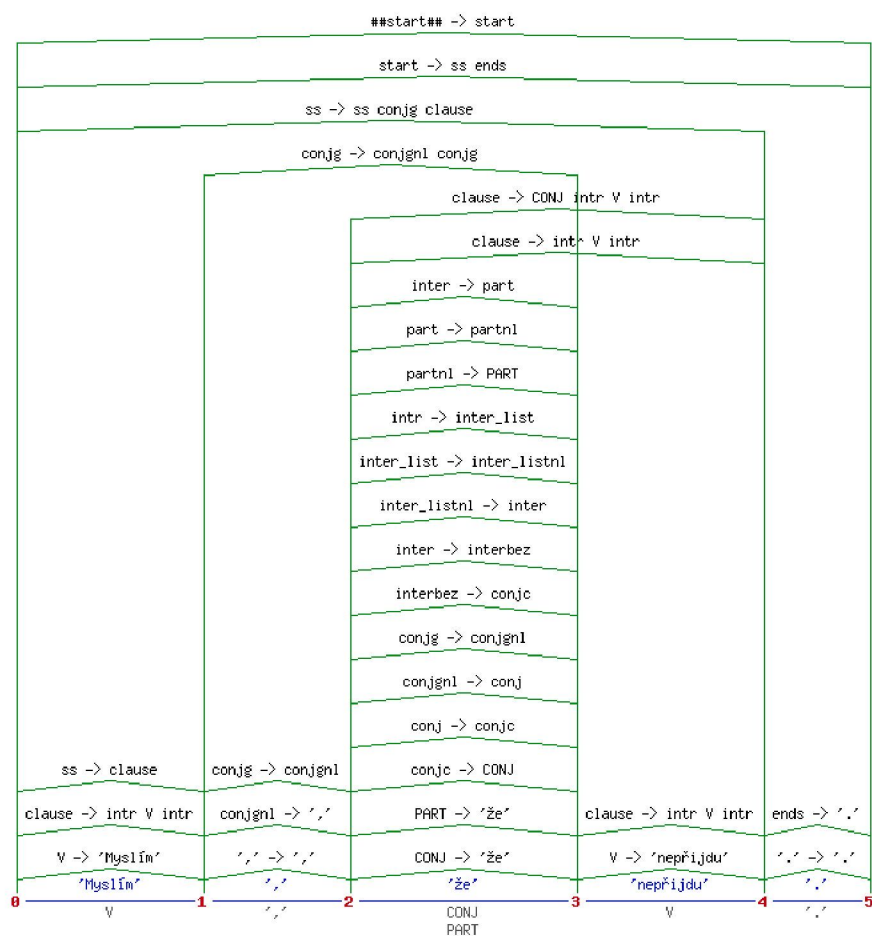
Literatura

- [Hau99] Hausser, Roland: *Foundations of Computational Linguistics*. Springer, Berlin Heidelberg, 1999, ISBN 3-540-66015-1.
- [HHK06] Hlaváčková, Dana, Horák, Aleš a Kadlec, Vladimír: *Exploitation of the verbalex verb valency lexicon in the syntactic analysis of czech*. In *Proceedings of Text, Speech and Dialogue 2006*, strana 85–92, Brno, 2006. Springer-Verlag.
- [HKK⁺01] Hajič, Jan, Krbec, Pavel, Květoň, Pavel, Oliva, Karel a Petkevič, Vladimír: *Serial combination of rules and statistics: a case study in Czech tagging*. In *ACL '01: Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, strana 268–275, Morristown, NJ, USA, 2001. Association for Computational Linguistics.
- [Hor01] Horák, Aleš: *The Normal Translation Algorithm in Transparent Intensional Logic for Czech*. Doktorská práce, Fakulta informatiky, Masarykova univerzita, Brno, listopad 2001.
- [Kad07] Kadlec, Vladimír: *Syntactic analysis of natural languages based on context-free grammar backbone*. Doktorská práce, Fakulta informatiky, Masarykova univerzita, Brno, září 2007.
- [KH05] Kadlec, Vladimír a Horák, Aleš: *New Meta-grammar Constructs in Czech Language Parser synt*. In *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2005.
- [KH07] Kovář, Vojtěch a Horák, Aleš: *Reducing the Number of Resulting Parsing Trees for the Czech Language Using the Beautified Chart Method*. In *Proceedings of 3rd Language and Technology Conference. Poznań*, strana 433–437. Wydawnictwo Poznańskie, 2007, ISBN 978-83-7177-407-2.
- [KHKar] Kovář, Vojtěch, Horák, Aleš a Kadlec, Vladimír: *New Methods for Pruning and Ordering of Syntax Parsing Trees: Comparison*

- and Combination*. In *Proceedings of Text, Speech and Dialogue 2008*. Springer-Verlag, to appear.
- [Kou05] Koukolík, František: *Mozek a jeho duše*. třetí, rozšířené a přepracované vydání. Galén, Praha, makropulos edice, 2005.
- [Pal08] Pala, Karel: *Jazykové inženýrství, počítače a přirozený jazyk*. muni.cz, únor 2008, ISSN 1801-0814.
- [PMZ89] Pala, Karel, Materna, Pavel a Zlatuška, Jiří: *Logická analýza přirozeného jazyka*. Academia, Praha, 1989, ISBN 80-200-0027-5.
- [PO92] Pala, Karel a Osolobě, Klára: *Základy počítačové lingvistiky*. Masarykova univerzita, Brno, 1992, ISBN 80-210-0341-3.
- [PRS97] Pala, Karel, Rychlý, Pavel a Smrž, Pavel: *DESAM — Annotated Corpus for Czech*. In *Proceedings of SOFSEM'97*, strana 523–530. Springer-Verlag, 1997.
- [Ryc00] Rychlý, Pavel: *Korpusové manažery a jejich efektivní implementace*. Doktorská práce, Fakulta informatiky, Masarykova univerzita, Brno, únor 2000.
- [Sed05] Sedláček, Radek: *Morphemic Analyser for Czech*. Doktorská práce, Fakulta informatiky, Masarykova univerzita, Brno, 2005.
- [Šm04] Šmerk, Pavel: *Unsupervised Learning of Rules for Morphological Disambiguation*. In *Lecture Notes in Artificial Intelligence 3206, Proceedings of Text, Speech and Dialogue 2004*, strana 211–216, Berlin, 2004. Springer-Verlag.

Dodatek A

Příklad výsledného chartu



Obrázek A.1: Příklad výsledného chartu vygenerovaného prostřednictvím webového rozhraní **wwwsynt** (autorem J. Golembiovský) pro vstupní větu: „*Myslím, že nepřijdu.*“

Dodatek B

Úplný výčet použitých funkcí

```
/*
 * Main function for structures extraction
 * params:
 * Chart *chart -- a-pointer to the output chart
 * WordList *originalWords -- a-pointer to the input wordlist (word-lemma-tag) structure
 */
void extras(Chart *chart, WordList *words);

/*
 * This function processes one edge for structures extraction
 * params:
 * Edge *edge -- edge to be processed
 */
void extras_edge(Edge *edge);

/*
 * Prunes the value according to morphological values which is to be backpropagated
 * params:
 * Value *upperValue -- value with morphological information to be backpropagated
 * Value *lowerValue -- value to be pruned according to upperValue
 */
void prune_values(Value *upperValue, Value *lowerValue);

/*
 * Uses breadth-first search in the forest of values and backpropagates morphological values
 * params
 * Value *Value -- current value to be backpropagated to its children
 */
void propagate_values_to_leaves_bfs(Value * value);

/*
 * Uses depth-first search in the forest of values and backpropagates morphological values
 * params
 * Value *upperValue -- current value to be backpropagated to its children
 */
void propagate_values_to_leaves_dfs(Value * value);

/*
 * Prunes the word structures according to the morphological value
 * params:
 * Value *value -- morphological value (from the preterminal edge PRE -> word)
 * Word *originalWord -- the input word structure (with input values from ajka)
 * Word *valuedWord -- the output word structure
 */
void prune_words_values(Value *value, Word *originalWord, Word *valuedWord);
```

```

/*
 * The 'large' heuristic for structure extraction
 * params:
 * Edge *edge -- edge to be procesed
 */
void extras_large(Edge *edge);

/*
 * The 'small' heuristic for structure extraction
 * params:
 * int left, right -- left/right boundaries of the edge to be procesed
 */
void extras_small(int left, int right);

/*
 * Prints the resulting structures
 */
void print_structures();

/*
 * Print ALL possible structure (i.e. edges with the given nonterminal on their left side)
 */
void print_all_structures();

/*
 * Removes any structures inner to [left,right)
 * params:
 * int left -- left boundary (including)
 * int right -- right boundary (not including)
 */
void remove_inner_structures(int left, int right);

/*
 * Splits the structure into extras parts according to case-number-gender agreement.
 *
 * params:
 * int structure -- structure to be considered whether or not to be splitted
 */
void split_structure_by_cng(int structure);

/*
 * Splits structure starting at 'structure' at position 'position'
 * (both structure and position are word list indeces)
 * params:
 * int structure -- structure to be splitted
 * int position -- split position
 */
void split_structure(int structure, int position);

/*
 * Checks for case-number-gender agreement
 * params:
 * morf_info *morfInfo1, morf_info morfInfo2 -- morphological values to be compared.
 * returns: 0 (false) / 1 (true)
 */
int agree_gncp(morf_info *morfInfo1, morf_info *morfInfo2);

```

```

/*
 * Splits the structure into extras parts according to the positions of prepositions
 *
 * params:
 * int structure -- structure to be considered whether or not to be splitted
 */
void split_structure_by_preposition(int structure);

/*
 * Prunes cases (except one) in the value
 * params:
 * int exceptCase -- the case which should be kept
 * Value *lowerValue -- the value in which the cases will be pruned
 */
void prune_case(int exceptCase, Value *lowerValue);

/*
 * Pushes a~new value at the end of the queue.
 * params
 * Value *value -- value to be pushed at the end of the queue.
 */
void queue_push(Value *value);

/*
 * Pops the value at the beginning of the queue.
 * returns: the popped value.
 */
Value* queue_pop();

/*
 * Checks queue consistency. The queue is used in propagate_values_bfs().
 * Queue consistency means that the invariant
 * "first->next1->next2->...->nextN(->NULL)", "nextN = last" holds.
 */
void queue_check_consistency();

/*
 * Checks whether this structure is entailed in the allStructures field.
 * If so, it is added to the structures field, otherwise the given position is cleared.
 * params:
 * int left, right -- structure identified by [left, right) boundaries to be included.
 */
void addStructure(int left, int right);

/*
 * Checks whether there is a~structures starting at the given index in the allStructures field.
 * params:
 * int index -- starting index
 * returns: 0 if no structure has been found, structure's end otherwise.
 */
int startsStructureAt(int index);

```

Dodatek C

Volby syntu relevantní pro extrakci struktur

Níže uvádím některé z dostupných parametrů pro spuštění syntaktického analyzátoru **synt**, které souvisí s extrakcí struktur. Úplný výčet voleb lze získat zadáním parametru **-h**.

Usage:

```
synt [-i plain | brief | vertical] [OPTIONS] < file.txt
```

OPTIONS:

```
-l num      ... use rules of level num
-i inp      ... set input type (plain, corpus, brief or vertical)
-pl         ... prune chart by rule levels
-n nterm    ... print structures for nonterminal nterm (use 'vp' for verbal structures)
-m          ... split structures by case-number-gender agreement (requires "-n nterm")
-s          ... look for smallest possible structures (default = largest) (requires "-n nterm")
-ps         ... print all structures (requires "-n nterm")
```