Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE

Ján Pecsők

Vzájomné odkazovanie slov v texte

Ústav formální a aplikované lingvistiky

Vedúci bakalárskej práce: Mgr. Barbora Vidová Hladká Ph.D.

Študijny program: Informatika, obecná informatika

2011

Ďakujem pánu Schlesingerovi a pani Hladkej za trpezlivosť a pomoc pri písaní tejto práce.

Prehlasujem že som svoju bakalársku prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím zo zapožičaním práce a jej zverejňovaním.

V Prahe dňa 29.7.2011

Obsah

Zoznam obrázkov

Zoznam textových polí

Názov práce: Vzájomné odkazovanie slov v texte

Autor: Ján Pecsők

Katedra: Ústav formální a aplikované lingvistiky

Vedúci bakalárskej práce: Mgr. Barbora Vidová Hladká Ph.D.

E-mail vedúceho: hladka@ufal.mff.cuni.cz

Abstrakt: Cieľom tejto bakalárskej práce je preskúmať možnosti hľadania koreferencií pomocou systému pravidiel na základe morfologických a syntaktických informácií. Súčasťou práce je aj vizualizácia koreferencií v texte a evaluácia jednotlivých pravidiel. Za týmto účelom bola vytvorená aplikácia Koreferencie, ktorá tvorí prostredie pre vizualizáciu textu, tvorenie a evaluáciu pravidiel. Vytvorená a evaluovaná bola sada pravidiel. Súčasťou práce je popis pravidiel a možnosti pravidlového systému spolu s algoritmom aplikácie pravidiel na text. Poslednú časť práce tvorí užívateľská a programátorská dokumentácia.

Kľúčové slová: Koreferencie, PDT, pravidlový prístup, automatické hľadanie

Title: Coreference in Text

Author: Ján Pecsők

Department: Institute of Formal and Applied Linguistics

Supervisor: Mgr. Barbora Vidová Hladká Ph.D.

Supervisor's email address: hladka@ufal.mff.cuni.cz

Abstract: The goal of this bachelor thesis is to explore possibilities of searching coreference with rule-based approach on the basis of morphological and syntactical information. Visualization of coreference as well as rule evaluation is part of the thesis. Application Koreferencie was developed for this purpose. It provide environment for visualization of text, creation and evaluation of rules. Set of rules was created and evaluated. Part of the thesis describes rules and application of rules on text. In the last part there is user and programmer documentation.

Keywords: Coreference, Rule-based approach, PDT, automatic resolution

1.Úvod

1.1 Koreferencie

Nasledujúci text je voľne preložený a doplnený text používaný[6] na osvetlenie pojmu koreferencie. Podrobnejšie o koreferenciach je možné nájsť v [8,10,11,12].

Príklad: **Otec** vždycky tvrdil, že opery nesnáší. Říkal, že **mu** na opeře vadí hlavně ten zpěv.

Slová otec a mu označujú jednu osobu, ktorá je otcom hlavného hrdinu vo vybranej knihe. Obidve slová sa zúčastňujú procesu, tzv. referencie, keď rečník používa rôzne výrazy, aby sa odkázal k osobám, predmetom alebo situáciám reálneho sveta. Výraz v prirodzenom jazyku, ktorý hrá úlohu odkazu, sa nazýva odkazujúci, a subjekt, na ktorý je odkazované, je odkazovaný. Teda otec a mu sú odkazujúce výrazy a pán otec v reálnom svete je ich odkazovaný. Vzťah medzi dvomi odkazujúcimi výrazmi je tzv. koreferencia. (Väčšinou) druhý (v texte uvedený) výraz v koreferenčnej dvojici (mu) sa nazýva anafora, prvý (v textu uvedený skôr) výraz (napríklad otec) je antecedent. Koreferencie majú dôležitú úlohu v mnohých oblastiach lingvistického sveta, ako napríklad

- automatické porozumenie textu
- strojové učenie
- automatický preklad
- dialógové systémy, automatické excerpcia informácie (information extraction);
- automatické odpovedanie na otázky (question answering)

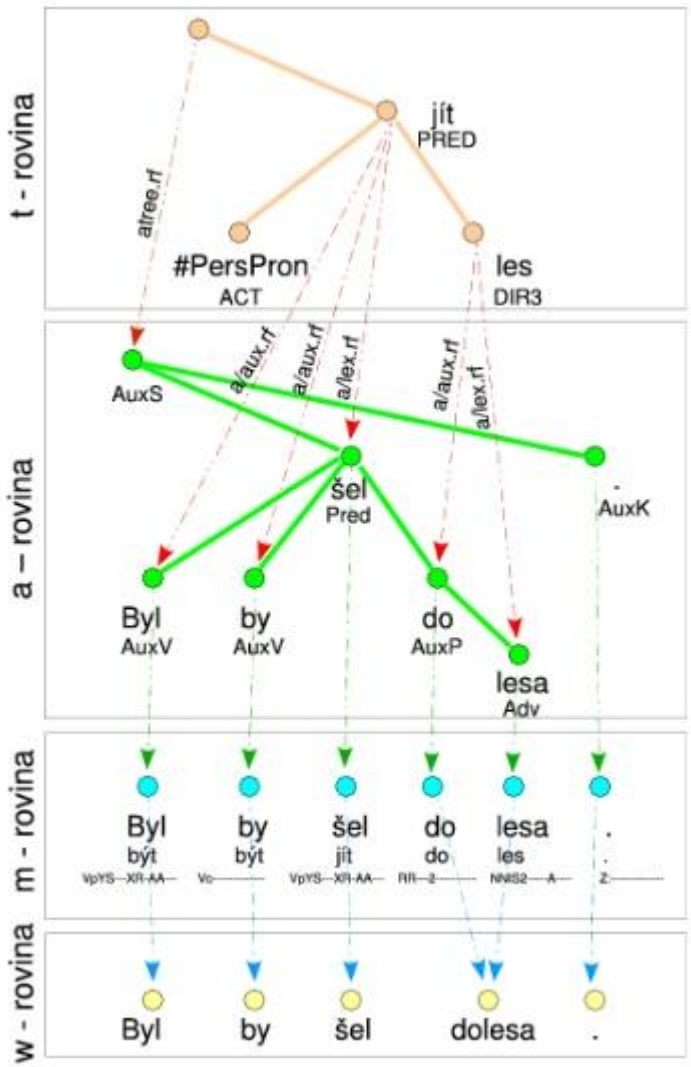
1.2 Pražský závislostný korpus (PDT)

Informácie o PDT sú k dispozícií na stránkach[9], z tohto zdroja preberám aj nasledujúce základné informácie.

Pražský závislostný korpus 2.0 (PDT 2.0) obsahuje veľké množstvo českých textov doplnených rozsiahlou a previazanou morfologickou (2 milióny slovných jednotiek), syntaktickou (1,5 miliónu slovných jednotiek) a sémantickou (0,8 miliónu slovných jednotiek) anotacií; na sémantické rovine sú navyše anotované aktuálne členenia vety a koreferenčné vzťahy. PDT 2.0 vychádza z dlhodobej pražskej lingvistickej tradicií, upravené pre súčasné potreby výskumu v oblasti počitačovej lingvistiky. Samotný korpus využíva najnovšie anotačné technológie. K dispozicií sú tiež softwarové nástroje pre prehľadávanie korpusu, anotáciu dát a jazykovou analýzu. Nechýba ani rozsiahla dokumentácia (v angličtině).

1.2.1 Roviny anotácie

Data v PDT 2.0 sú anotované na troch rovinách: na *morfologickej rovine*, *analytickej rovine* a *tektogramatickej rovine*. V skutočnosti existuje ešte jedna, neanotačná rovina, reprezentujúca "surový text". Na tejto rovine, zvanej *slovná rovina*, je text rozdelený do dokumentov a odstavcov. Sú tu rozlíšené slovné jednotky (slova, čísla, interpunkcia) a sú opatrené jednoznačnými identifikátormi. Slovná rovina je nazývaná tiež *w-rovina*, morfologická *m-rovina*, analytická *a-rovina* a tektogramatická *t-rovina*. Podobne je uzol stromu reprezentujúci analytickú anotáciu vety nazývaný *a-uzol* atď. Príklad prepojenia jednotlivých rovín ilustruje obrázok 1 na príklade českej vety *Byl by šel dolesa.*



Obrázok 1 Prepojenie rovín textu

V tejto práci sa zaoberám výlučne m-rovinou a a-rovinou. Hĺbková analytická t-rovina sa nepoužíva, na nej sú bežne anotované koreferenčné javy, pre potreby práce je však možné presunúť anotáciu koreferenčných párov na m-rovinu. Koreferencie a ďalšie javy sú v PDT z ručne anotované kvalifikovanými odborníkmi. Anotáciou koreferencií sa podrobnejšie zaoberajú práce [10,11,12]

2 Hľadanie pomocou pravidlového prístupu

Pri hľadaní koreferencií pomocou pravidiel bolo nutné vytvoriť dostatočne flexibilný pravidlový systém umožňujúci hľadanie dvojíc slov. Následne bolo potrebné vytvoriť postup ako aplikovať pravidlá na annotovaný text. Pokusy o automatizáciu hľadania koreferencií zahrňujú práce Lappin a Leass [14], ktorý však nepracujú zo závislostnými stromami. Na dátach PDT sú napríklad [7,13,15].

2.1Popis pravidlového systému

Výstupom aplikácie pravidla má byť množina dvojíc slov, ktoré na seba odkazujú. Pravidlo teda musí tiež mať spôsob ako na seba odkázať dvojicu slov. Pri tvorbe pravidlového systému bol rešpektovaný systém rovín jazyka. Pravidlo musí vedieť využiť informácie o jednotlivých slovách z morfologickej roviny aj z analytickej roviny a tiež musí vedieť brať do úvahy kontext v ktorom sa slová nachádzajú. Na morfologickej rovine má text lineárnu štruktúru, na analytickej rovine má stromovú štruktúru, túto skutočnosť musí pravidlový systém zohľadniť. Je potrebné určiť rozsah kontextu a vysporiadať sa s väčším množstvom nájdených dvojíc.

2.1.1 Myšlienka dvojice podpravidiel

Aby pravidlo hľadalo dvojice slov, každé pravidlo sa skladá z dvojice pravidiel. Prvé pravidlo hľadá tzv. Zdroj, to znamená prvé slovo z koreferenčného páru. Toto pravidlo sa aplikuje na celý text a výstupom je množina antecedentov, Zdrojov . Druhé pravidlo hľadá ku každému antecedentu anaforu, respektívu ku každému Zdroju Cieľ. Nerozlišujem presne, ktorý z týchto 2 sa nachádza ako prvý, to záleží na tom ako vyzerajú pravidlá, môžem si vytvoriť pravidlo, ktoré ku každej anafore hľadá antecedent. To sa aplikuje na časť textu v okolí každého Zdroja. Rozsah tohto okolia je súčasťou pravidla. Výstupom aplikovania obidvoch podpravidiel je množina dvojíc (Zdroj, Cieľ).

2.1.2 Syntax a význam pravidiel

Podmienky vlastnosti slova

Podmienky vlastnosti slova sú logické výrazy pre ktoré je vstupom české slovo s anotovanými vlastnosťami morfologickej a syntaktickej roviny a výstupom je pravdivostná hodnota **true** alebo **false**, teda či dané slovo vyhovuje podmienke. Príklad prázdnej podmienky, ktorej výstup je vždy true: []

Vlastnosti a porovnávacie operátory

Vlastnosti na ktoré sa môžeme pýtať sú tieto:

- Afun - analytická funkcia. Jednotlivé hodnoty sú v prílohe.
- Tag - Morfologická značka (tag), tu je možné sa pýtať aj na jednotlivé zložky tag.1,2 až 15 Jednotlivé hodnoty v prílohe.
- Lemma - Základný tvar slova. Napríklad jednotné číslo, prvý pád pre podstatné mená. Neurčitok pre slovesá a podobne.
- Form - Aktuálny tvar slova v texte.
- Id - Jednoznačne priradený kód slova v texte.

Porovnávacie operátory sú „==“ a “!=”. Porovnávať je možné aj s regulárnymi výrazmi. Príklady jednoduchých podmienok

- [afun=="Atr"] - Slovo ktoré má hodnotu analytickej funkcie attribute.
- [tag.2=="D"] - Slovo ktoré má hodnotu tag na 2 mieste “D”.
- [lemma=="který"] - Slovo, ktorého základný tvar je „který“.
- [lemma=="[A-Z].*"] -Slovo, ktorého základný tvar začína na veľké písmeno od A do Z.
- [form=="\."] -Slovo označujúce bodku. Lomítko je potrebné, pretože bodka má význam v regulárnych výrazoch. Viac o regulárnych výrazoch v prílohe.

Logické operátory

Podmienky je možné spájať pomocou logických operátorov

- && - operátor AND príklad: [afun!="Atr" && tag.2!="D"]
- || - operátor OR príklad: [tag.3=="Z" || tag.3!="F"]

- ! - operátor negácie príklad: [!tag.3=="Z"]

2.1.3Určenie kontextu v lineárnej štruktúre textu

K určenie kontextu slov, tak Zdroja ako aj Cieľa v lineárnej štruktúre textu na morfologickej rovine sú použité regulárne výrazy. Namiesto bežných regulárnych výrazov, ktoré fungujú na písmenách, fungujú v tomto pravidlovom systéme regulárne výrazy na slovách. Ak neuvažujeme žiaden kontext iba vlastnosti Zdroja a cieľa, potom pravidlo je v tvare.

[**Zdroj** <Podmienka vlastností slova>] [**Ciel** <Podmienka vlastností slova>]

Príklad 1

Zdroj a Ciel sú povinné premenné, podrobne o premenných je napísané ďalej v texte. Budú však uvádzané v niektorých príkladoch aby si čitateľ zvykol.

„.“-Operátor zreťazenia. Najčastejšie používaný operátor. Týmto môžeme za sebou zoradiť výrazy. Obecne *výraz1.výraz2* je výraz v ktorom text splňujúci *výraz1* predchádza textu splňujúci *výraz2*.

V príklade 2 je výraz, ktorý nájde dve za sebou idúce slová, z ktorých druhé je podstatné meno a prvé vyjadruje nejakú vlastnosť tohto slova. Podstatné meno sa uloží do premennej Zdroj.

[afun=="Atr"].[Zdroj tag.1=="N"]

Príklad 2

...kraje, země či regiony budou hospodařit podle rozpočtu ... ...pravomoci vytýčil celkům, jejichž počet je dodnes ve hvězdách... Nebude žádné zbídačení mas, žádné imperialistické koloniální války.

Text 1 Príklad českého textu

Vyhovujúce slová: *Poklidné* ***kompetence****, Miroslav* ***Korecký****, samosprávných* ***celků****, Tato* ***úprava***

Zvýraznené slovo z dvojice sa uloží do premennej Zdroj

(Výraz)* - Operátor hviezdička. Znamená, že predchádzajúci výraz sa môže nula alebo viackrát opakovať. V nasledujúcom príklade 3 je výraz, ktorý vyhovuje sérií slov, ktoré začínajú „**,**“- čiarkou pokračujú žiadnymi alebo niekoľkými atribútmi a posledným je podstatné meno. Do premennej Zdroj sa uloží práve toto posledné slovo.

```
[form==","].[afun=="Atr"]*.[Zdroj tag.1=="N"]
```

Príklad 3

Walrase pokládá mimochodem za největšího ekonoma všech dob - a pojímá jej jako ryze stacionární.Dočasný podnikatelův zisk bude anulován, ale trvalý zisk z jeho inovace zůstane zachován společnosti ve formě nižších cen výrobků.

Text 2 Príklad českého textu

vyhovujúce slovné spojenia v texte 2

- ***, země,***
- , jejichž **počet,**
- , žádné imperialistické koloniální **války**

(Výraz1) | (Výraz2) - Operátor pipe. Binárny operátor, ktorý povoľuje jeden z dvoch výrazov. Obecne *výraz1|výraz2* vyhovuje text, ktorý vyhovuje výrazu 1 alebo výrazu 2. V nasledujúcom príklade 4 je výraz ktorý vyhovuje slovu „jeho“ alebo “jej” pričom nasledujúce slovo vloží do premennej Zdroj.

```
[form==","].[afun=="Atr"]*.[Zdroj tag.1=="N"]
```

Príklad 4

V texte 2 sú vyhovujúce slová: *jej **jako**, jeho **inovace***

(Výraz)+ - Operátor plus. Znamená že predchádzajúci výraz sa môže raz alebo viackrát opakovať. Oproti operátoru * sa líši nutnosťou aspoň jedného opakovania. V upravenom príklade 3

```
[form==","].[afun=="Atr"]+.[Zdroj tag.1=="N"]
```

Príklad 5

A v texte 3 vyhovujúce len spojenia

- *, jejichž **počet***
- , žádné imperialistické koloniální **války**

(Výraz)? - Operátor otáznik. Znamená že predchádzajúci výraz sa neopakuje alebo sa objaví práve raz. Použijeme upravený príklad 3

```
[form==","].[afun=="Atr"]?.[Zdroj tag.1=="N"]
```

Príklad 6

Vyhovujúce spojenia v texte 2

- ***, země***
- , jejichž **počet**

Podmienka na počet nájdených koreferencií

Pravidlo často môže nájsť viac ako jeden Cieľ k Zdroju. My však potrebujeme väčšinou jeden, maximálne niekoľko Cieľov ku každému Zdroju. Preto súčasťou pravidla je obmedzenie na maximálny počet nájdených Cieľov pre jeden Zdroj. Spôsob prioritizácie je jednoduchý, vyberú sa najprv slová, ktoré sú najbližšie zľava od Zdroja, potom najbližšie sprava.

Podmienka na šírku cieľového kontextu

Často nepotrebujeme prehľadávať celý text pre každý Zdroj, stačí nám nejaké jeho najbližšie okolie preto súčasťou pravidla je určenie hraníc okolia. Hranice sú určené dvomi číslami. Prvé z nich určuje od ktorého slova zľava od Zdroja začína okolie. Je to záporné číslo alebo 0. Druhé číslo je nezáporné a určuje po ktoré slovo sprava od Zdroja končí okolie.

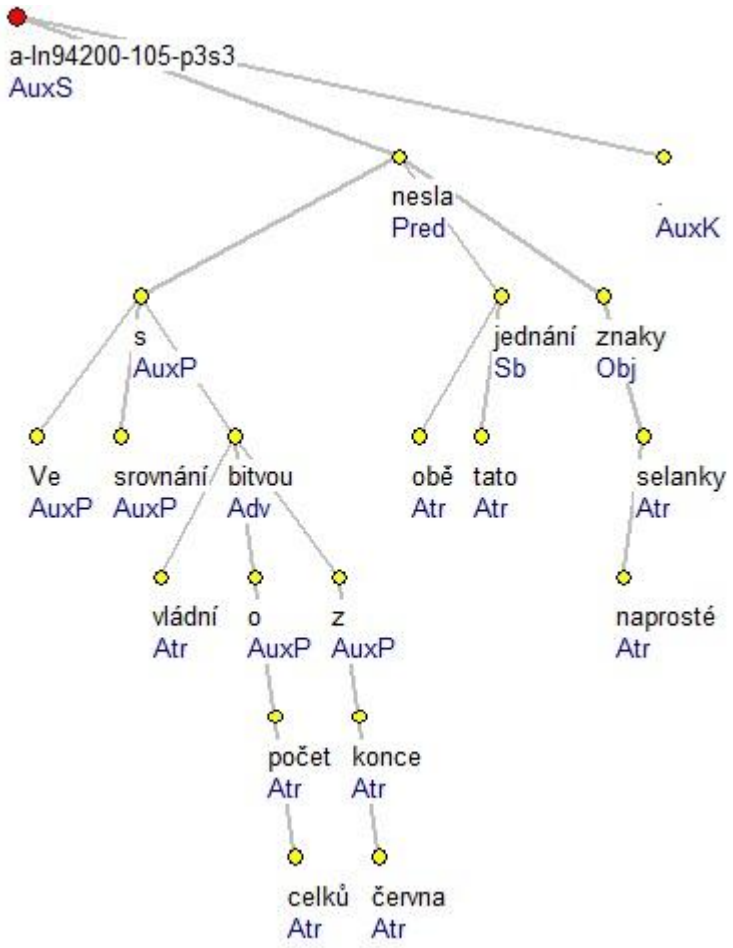
2.1.4 Určenie kontextu v stromovej štruktúre textu

Pri písaní pravidiel je tiež užitočné určiť kontext v analytickej rovine textu. Tá má stromovú štruktúru. Aby bolo možné sa pýtať aj na túto rovinu, slúžia vlastnosti slova **parent**, **children** a **ancestor**.

Parent funguje ako otázka na vlastnosti otca v analytickej rovine textu. Obrázok 1. ukazuje text 4 v analytickej rovine a anotovanou hodnotou analytickej funkcie.

Ve srovnání s vládní bitvou o počet celků z konce června nesla obě tato jednání znaky naprosté selanky.

Text 3 Príklad českej vety



Obrázok 2 Analytická rovina textu 4

Nasledujúci príklad 7 vyhovuje pre slová ktoré majú rodiča s analytickou funkciou attribute. Výraz **parent** musí byť napísaný malými písmenami.

```
[Zdroj parent< afun=="Atr" >]
```

Príklad 7

Vyhovujúce slová: *celkú, června, naprosté*

Pravidlá je možné vkladať do seba. Ďalšiemu výrazu vyhovujú slová, ktoré majú rodiča s anal. funkciou Atr a prarodiča s analytickou funkciou AuxP.

```
[Zdroj parent< afun=="Atr" && parent< afun=="AuxP">>]
```

Príklad 8

Vyhovujúce slová v texte 4 sú: *celkú, června*

Ďalšou možnosťou je použiť dotaz **ancestor**, ktorý je pravdivý ak jeho podmienka je pravdivá pre niektorého predka v analytickej rovine textu. V príklade 9 vyhovuje výraz slovám, ktoré majú niektorého predka s analytickou funkciou rovnou „Adv“.

```
[Zdroj ancestor< afun=="Adv">]
```

Príklad 9

Vyhovujúce slová v texte 4 sú: *vládní, o, počet, celků, z, konce, června*

Posledný druh dotazu na kontext v analytickej rovine je **children**. Ten vyhovuje vtedy, keď aspoň jedno slovo vyhovuje podmienke. Výrazu v príklade 10 vyhovujú slová, ktoré majú aspoň jedného priameho potomka splňujúceho afun=="Atr".

```
[Zdroj children< afun=="Atr" >]
```

Príklad 10

Vyhovujúce slová v texte 4 sú: *bitvou, o, počet, z, konce, jednání, znaky, selanky*

2.1.5 Použitie premenných

Tak ako bolo uvedené vyššie, každé pravidlo má dve časti. Aby sme mohli prenášať informácie z pravidla pre Zdroj do pravidla pre Cieľ, chceme napríklad aby Zdroj a Cieľ boli v rovnakom rode. Potrebujeme použiť premenné, do ktorých si uložíme

informácie o slove. Štandardné premenné sú **Zdroj** a **Ciel**, ktoré **musia byť** v každom pravidle. Premenná je vždy prvé slovo oddelené medzerou od zvyšku podmienky.

```
Pravidlo pre Zdroj   [Zdroj lemma=="[A-Z].*"]

Pravidlo pre Cieľ    [Ciel lemma==Zdroj.lemma && id!=Zdroj.id]
```

Príklad 11

Pravidlo hľadá ku každému zdroju slová ktoré sa zhodujú v lemme (základný tvar slova), nezhodujú sa však v identifikačnom čísle, takže to nie sú tie isté slová. Na obsahy premenných sa pýtame <meno premennej>.<atribút>.V príklade 12 máme pravidlo pre Zdroj, ktoré hovorí aby každé podstatné meno pred ktorým je slovo s hodnotou analytickej funkcie (afun) „*attribute*“ bolo označené ako Zdroj.

```
[afun=="Atr"].[Zdroj tag.1=="N"]
```

Príklad 12

Ak by sme upravili pravidlo nasledujúcim spôsobom.

```
[Zdroj afun=="Atr"].[tag.1=="N"]
```

Príklad 13

Potom ako Zdroj bude označené slovo s afun=”atr”, ktoré predchádza nejakému podstatnému menu. Ak chceme môžeme použiť premenné aj v podmienkach pre dotazy **parent, ancestor** a **children**. V nasledujúcom príklade je pravidlo, ktoré pre všetky slová, ktoré majú základ „který“, nájde prvého predka v strome na analytickej rovine, ktorý spĺňa určité podmienky. Tohto predka označí za Cieľ.

Pravidlo pre Zdroj: [Zdroj lemma=="který"]

Pravidlo pre Cieľ: [id==Zdroj.id && ancestor<Ciel Zdroj.tag.3==tag.3 && Zdroj.tag.4==tag.4 && tag.1!="V">]

Príklad 14

Okrem štandardných premenných, ktoré musia byť v každom pravidle, je možné použiť aj ďalšie pomocné premenné. V nasledujúcom jednoduchom príklade pravidlo pre Zdroj nájde prídavné meno (tag.1=="A" ako Adjective) ktoré predchádza podstatnému menu (tag.1=="N" ako Noun). Prídavné meno sa uloží do premennej Zdroj a podstatné meno sa uloží do premennej T. V pravidle pre Cieľ potom môžeme použiť premennú T, v tomto prípade v podstate len za Ciel označíme slovo ktoré bolo v premennej T.

Pravidlo pre Zdroj: [Zdroj lemma=="který"]

Pravidlo pre Cieľ: [id==Zdroj.id && ancestor<Ciel Zdroj.tag.3==tag.3 && Zdroj.tag.4==tag.4 && tag.1!="V">]

Príklad 15

2.1.6 Zložené pravidlo

Doteraz bola popísaná syntax jednoduchých pravidiel. Z týchto pravidiel je možné tvoriť pravidlá zložené pomocou binárnych množinových operátorov. Tie sa totiž aplikujú na množiny dvojíc, ktoré vzniknú aplikovaním jednoduchých pravidiel. Existujú 3 druhy operátorov. Je možné zložiť medzi sebou ľubovoľne zložené a jednoduché pravidlá. Čím vzniká binárny strom, ktorý má vo vnútorných uzloch binárne operátory a v listoch jednoduché pravidlá. Všetky uzly majú svoje názvy. Operátory

- || - Operátor zjednotenia. Dvojica slov patrí do výsledku ak patrí aspoň do jedného pravidla.
- **&&-** Operátor prieniku. Dvojica slov patrí do výsledku ak patrí do obidvoch pravidiel.
- **A/B-** Operátor mínus. Dvojica slov patrí do výsledku ak patrí do množiny výsledku pravidla A, ale nepatrí do výsledku pravidla B.

Na obrázku 2 je vidieť zložené pravidlo, ktoré vzniklo zjednotením jednoduchých pravidiel *PravidloZamena1-29* a *PravidloLemma* čím vzniklo zložené pravidlo *ZamenaLemma*, ktoré následne sa opäť zjednotením spojilo s pravidlom *PravidloKtery1-2.1*.

Obrázok 3 Strom zloženého pravidla

2.1.7 Program Bonito a prehľad dotazovacích jazykov

Program riešiaci podobný problém som nenašiel. Nechal som sa však inšpirovať programom Bonito, ktorý používa podobné regulárne výrazy na slovách. Viac informácií je na stránkach [18,19]. Príklady ďalších dotazovacích jazykov

2.2 Popis spôsobu aplikovania pravidiel

Po vypracovaní pravidlového prístupu, je potrebné navrhnúť spôsob aplikácie pravidiel na text. Keďže pravidlový systém vychádza z regulárnych výrazov, ktoré sa aplikujú pomocou nedeterministických konečných automatov, bola použitá ich modifikácia. V tejto kapitole popíšem konštrukciu automatu, aké má súčasti a ako sa

aplikuje na text. Príslušným materiálom je akýkoľvek zdroj informácií o automatoch a gramatikách napríklad [2],[3].

2.2.1 Konečné automaty

Nasledujúci text preberám z väčšej časti so zdroju [4], jedná sa o základné definície.

Definícia DKA

Deterministický konečný automat je pätica (Σ,*K*,q_0,δ,*F*), kde:

- Σ je vstupná abeceda (neprázdna konečná množina symbolov).
- *K* je konečná množina stavov.
- q_0 je počiatočný stav, pričom platí $q_0 \in K$.
- δ je prechodová funkcia: $\delta : K \times \Sigma \to K$, čiže funkcia, ktorá na základe stavu a symbolu zo vstupnej abecedy vráti nový stav
- *F* je množina akceptačných stavov, je to ľubovoľná (môže byť aj prázdna) podmnožina *K*. Hovoríme, že DKA akceptuje slovo *w*, ak výpočet na tomto slove skončí v niektorom z akceptačných stavov.

Konfigurácia DKA

Konfigurácia deterministického konečného automatu je dvojica

$(q, w) \in K \times \Sigma^*$, kde *q* je aktuálny stav automatu a *w* je dosiaľ neprečítaná časť vstupného slova.

Krok výpočtu DKA

Krok výpočtu deterministického konečného automatu je relácia $\vdash_A$na konfiguráciách DKA definovaná nasledovne: $(q, av) \vdash_A (p, v) \iff p = \delta(q, a)$.

Pod výpočtom na deterministickom konečnom automate rozumieme ľubovoľnú postupnosť na seba nadväzujúcich výpočtových krokov.

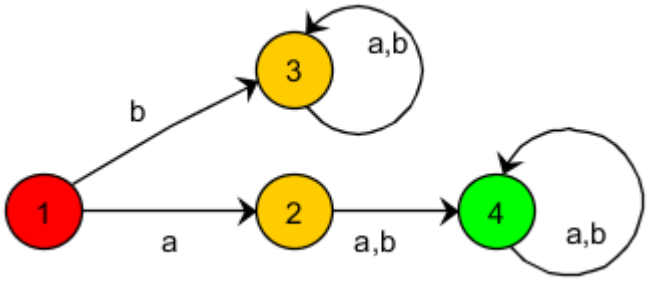
Jazyk akceptovaný pomocou DKA

Jazyk akceptovaný deterministickým konečným automatom *A* definujeme nasledovne:

$$L(A) = \{w \mid \exists p \in F : (q_0, w) \vdash_A^* (p, \varepsilon)\}.$$

Je to teda množina všetkých slov, na ktorých existuje v automate *A* výpočet končiaci v akceptačnom stave (takému výpočtu sa tiež hovorí akceptačný výpočet).

Na obrázku 3 vidieť príklad konečného automatu. Konkrétne tento prijíma v abecede {a,b} slová začínajúce na „a“. Jednotlivé uzly reprezentujú stav automatu. Šípky medzi uzlami reprezentujú prechodovú funkciu. Uzol 1 je počiatočný stav. Uzol 4 je akceptačný stav. V KA z každého uzlu je definovaná prechodová funkcia pre každý prvok abecedy a prechádza práve do jedného ďalšieho stavu.



Obrázok 4 Príklad deterministického konečného automatu

Nedeterministický konečný automat

Definícia NKA

Nedeterministický konečný automat je pätica (Σ,K,q_0,δ,F), kde:

- Σ je vstupná abeceda (neprázdna konečná množina symbolov).
- K je konečná množina stavov.
- q_0 je počiatočný stav, pričom platí $q_0 \in K$.
- δ je prechodová funkcia: $\delta : K \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^K$, čiže funkcia, ktorá na základe stavu a symbolu zo vstupnej abecedy množinu nových stavov
- F je množina akceptačných stavov, je to ľubovoľná (môže byť aj prázdna) podmnožina K. Hovoríme, že DKA akceptuje slovo w, ak výpočet na tomto slove skončí v niektorom z akceptačných stavov.

Existujú teda dva podstatné rozdiely medzi NKA a DKA:

- NKA povoľujú prechody na ε
- Nový stav nie je pre každý prechod určený jednoznačne. Prechodová funkcia vracia celú množinu stavov (pri výpočte sa môže postupovať do ľubovoľného z nich), ktorá môže byť dokonca prázdna.

Konfigurácia NKA

Konfigurácia nedeterministického konečného automatu sa definuje podobne ako pri deterministických konečných automatoch. Je to dvojica $(q, w) \in K \times \Sigma^*$, kde *q* je aktuálny stav automatu a *w* je dosiaľ neprečítaná časť vstupného slova.

Krok výpočtu NKA

Krok výpočtu je relácia \vdash_{A} na konfiguráciách NKA *A* definovaná nasledovne:

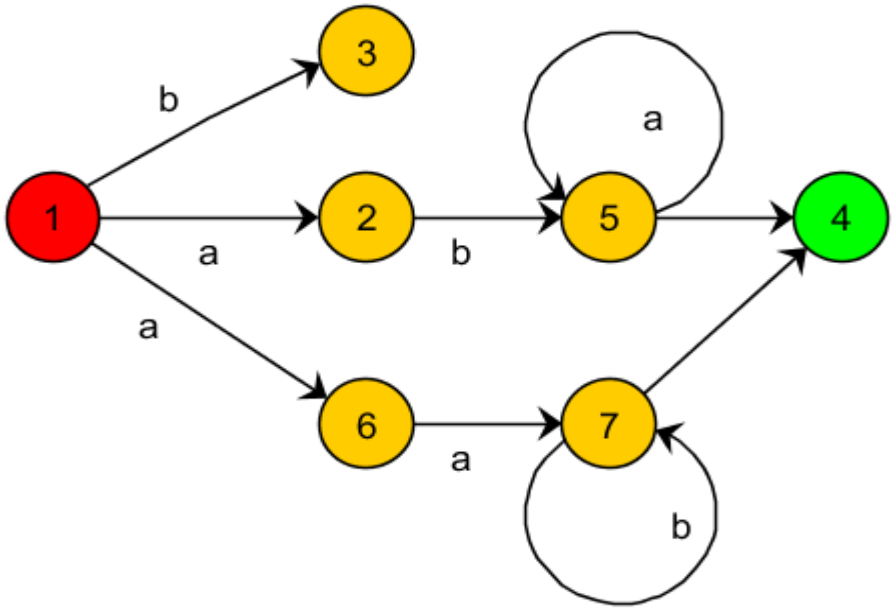
$$(q, aw) \vdash_A (p, w) \iff p \in \delta(p, a).$$

Jazyk akceptovaný pomocou NKA

Jazyk akceptovaný nedeterministickým konečným automatom *A* je množina

$$L(A) = \{w \mid \exists p \in F : (q_0, w) \vdash_A^* (p, \varepsilon)\}.$$

Na obrázku 4 je príklad NKA, ktorý prijíma slová v abecede {a,b}, ktoré začínajú na "ab" a pokračujú samými „a" alebo slová začínajúce na „aa" a pokračujúce samými „b". Na rozdiel od KA nemusí viesť od každého slova prechodová funkcia (stav 3), môže viesť len pre niektoré znaky z abecedy (v stave 2,6,5,7,4), alebo naopak môže z jedného stavu prejsť do viacerých stavov po jednom písmene (stav a). Taktiež môže obsahovať prázdne hrany, po ktorých je možné prejsť kedykoľvek bez vloženia písmena (hrany 5-4, 7-4).



Obrázok 5 Príklad nedeterministického konečného automatu

Ekvivalencia DKA a NKA

V skutočnosti, napriek rozdielnej definícii oboch výpočtových modelov, je ich výpočtová sila rovnaká. Je dokázané, že ku každému nedeterministickému automatu *A* existuje deterministický konečný automat *B* taký, že *L(B) = L(A)*. Opačná inklúzia je zrejmá z faktu, že deterministický automat je špeciálny prípad nedeterministického.

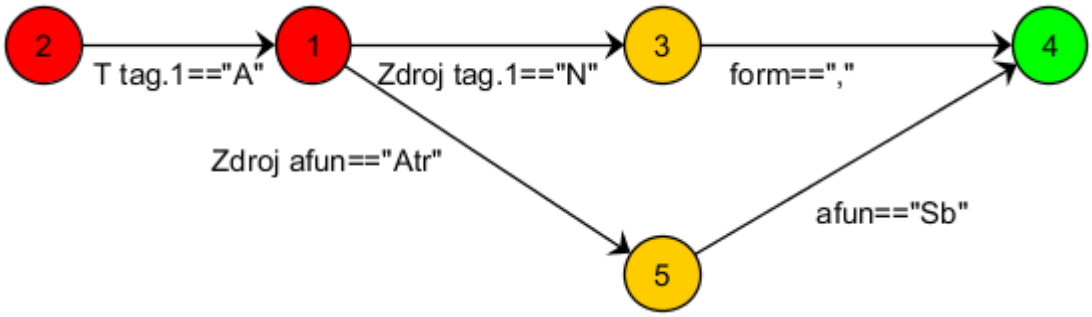
2.2.2 Automat použitý na aplikovanie pravidiel

Automat, ktorý je použitý na aplikáciu pravidiel je špecifický v tom, že jeho vstupná abeceda je množina slov českého jazyka (V podstate aj akéhokoľvek iného jazyka, to záleží od vstupných dát), ktorá zrejme nie je konečná a aj keby sme ju nejakým spôsobom obmedzili, stále je veľmi rozsiahla. Preto prechodová funkcia nie je definovaná osobitne pre každé jednotlivé slovo z tejto abecedy, skôr definuje podmienku vlastností slova, ktorú musí slovo splniť aby sa dostalo do ďalšieho stavu. Automat tiež pripúšťa viac iniciálnych stavov, po počiatočnej konštrukcií sú však odstránené všetky ε hrany.

Obrázok 5 ukazuje automat použitý na aplikovanie pravidla v príklade 16. Stavy 2 a 1 sú iniciálne stavy, stav 4 je koncový. V stave 1 je vidieť, že vychádzajú z neho 2 podmienky do rôznych stavov, podmienka tag.1==”N” a afun==”Atr”. Tieto podmienky sa navzájom nevylučujú a preto sa môže stať, že nejaké slovo spĺňa oboje. V tom prípade je potrebné preskúmať obidve možnosti, prejsť do obidvoch stavov. Tiež si všimnite premenné Zdroj a T, slovo sa uloží do premennej ak prejde hranou na ktorej sa premenná nachádza a premenná ešte žiadne slovo neobsahuje.

```
[T tag.1=="A"]?.(([Zdroj tag.1=="N"].[form==","])|
([Zdrojafun=="Atr"].[afun=="Sb"]))
```

Príklad 16



Obrázok 6 Automat použitý na aplikáciu pravidla z Príkladu 16

2.2.3Aplikácia automatu na text

V definícií NKA je, že automat prijíma slovo keď existuje postupnosť krokov výpočtu od iniciálneho stavu ku koncovému stavu po vložení všetkých písmen v slove. V našom prípade však pracujeme s celým textom a v ňom potrebujem úseky textu, ktoré dokáže automat prijať. Navyše, máme pravidlo skladajúce sa z dvoch podpravidiel vzájomne prepojenými pomocou premenných. K popisu použitého algoritmu potrebujeme nasledujúce dátové štruktúry.

- Automat- Dátová štruktúra zachycujú konfiguráciu automatu, musí v nej byť informácia o automate, v ktorom stave sa nachádza, obsah premenných.
- Premenné- Hodnoty premenných v automate, vrátane premenných Zdroj a Ciel

V texte 6 je pseudokód funkcie hľadania koreferencií v texte. Pseudokód funkcie AplikujAutomatNaText() je v prílohe A.

```
(Slovo,Zoznam Slov) NajdiKoreferencie(Zoznam Slov Text, PravidloZdroj, PravidloCiel, int
Kontext1, int Kontext2)
{
 (Slovo,Zoznam Slov) Koreferencie
 Zoznam Premenne Zdroje=AplikujAutomatNaText(Text, PravidloZdroj, null);
 foreach (Premenne P in Zdroje)
{
  if(P obsahuje Zdroj)
   {
      Zoznam Slov Kontext=VytvorKontext(Zdroj,Text,Kontext1,Kontext2);
      Ciele=AplikujAutomatNaText(Kontext,PravidloCiel,P);
      Koreferencie.add(Zdroj,Ciele);
   }
 }
 return Koreferencie
}
```

Text 5 Pseudokód hľadania koreferencií v texte

2.2.4 Aplikovanie zloženého pravidla

Ako bolo popísané vyššie, zložené pravidlo je spojením dvoch pravidiel pomocou niektorého množinového operátora. Preto aplikácia zloženého pravidla je realizovaná jednoduchou rekurziou, ktorej postup je vidieť v Texte 7.

```
AplikujPravidlo(Zoznam Slov Text ,Pravidlo P)
{
if(P je Zlozene Pravdlo)
 {
 P1=P.LavePravidlo;
 P2=P.PravePravidlo;
 Operator O=P.Operator
 Koref Dvojice V1= AplikujPravidlo(Text,P1);
 Koref Dvojice V2= AplikujPravidlo(Text,P2);
 return AplikujMnozinovyOperator(O,V1,V2)
 }else
 {
  return NajdiKoreferenci(Text,P.ZdrojPravidlo,
P.CielPravidlo,P.Kontext1,P.Kontext2);
 }
}
```

Text 6 Pseudokód aplikovania zloženého pravidla

V tejto časti popíšem konštrukciu automatu pre pravidlá. Pravidlo je vyhodnocované ako aritmetický výraz, pričom miesto čísel sú automaty a operátory "*","+","?",".","|" ktorých význam je popísaný v 2.1.2 . Najprv sa pravidlo prevedie zo základného infix tvaru do postfixového tvaru. Popis algoritmu je napríklad tu [4]. Príklad 16 po prevedení do postfixu by vyzeral nasledovne.

```
[T tag.1=="A"]?[Zdroj tag.1=="N"][form==","]. [Zdroj
afun=="Atr"][afun=="Sb"].|.
```

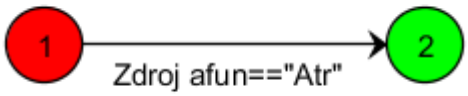
Príklad 17 Postfixový tvar Prikladu 16

Takto spracovaný výraz je pripravený na postupný prevod na automat. Postup prevodu je v texte 8.

```
Automat ZpracujPravidlo(Vyraz v Postfixu V){
Automat Operand1;
Automat Operand2;
Stack<Automat> Zasobnik;
while (Nie je koniec V)
{
 Token T=V.Next
 if (T je Podmienka vlastnosti slova)
 {
  Zasobnik.push(VytvorAutomatzPVS(T));
 }
 if (T je jeden z "*","+","?"){
   Operand1=Zasobnik.pop();
   Zasobnik.push(Operand1.AplikujOperator(T));
 }
 if (T je "." alebo "|" ))
 {
   Operand1=Zasobnik.pop();
   Operand2=Zasobnik.pop();
   Zasobnik.push(Operand1.AplikujOperator(T,Operand2));
 }
}
return Zasobnik.peek();
}
```

Text 7 Pseudokód spracovania výrazu v postfixovom tvare

Automat vytvorený z podmienky vlastnosti slova, je jednoduchý 2 stavový automat. Napríklad z podmienky [Zdroj afun==”Atr”] . Vznikne automat na obrázku 6.

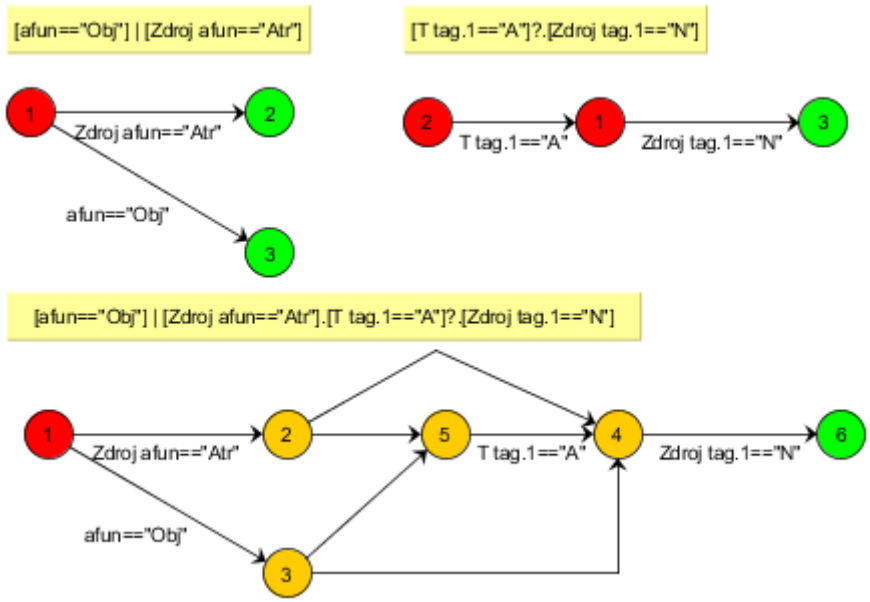


Obrázok 7 Automat vygenerovaný z podmienky vlastnosti slova

Jednotlivé operátory sa na automaty aplikujú nasledujúcim spôsobom. Význam operátorov je popísaný v časti 2.1.2.

- “.” – Operátor zreťazenia ako binárny operátor na automatoch sa aplikuje tak, že sa pridá prázdna hrana tzv. ε od každého koncového stavu prvého

automatu do každého iniciálneho stavu druhého automatu. Konečné stavy prvého automatu, prestanú byť konečnými stavmi. Príklad je na obrázku 7.



Obrázok 8 Príklad zreťazenia dvoch automatov

Formálnejí: $A = (\Sigma, K_1, Q_1, \delta_1, F_1), B = (\Sigma, K_2, Q_2, \delta_2, F_2)$

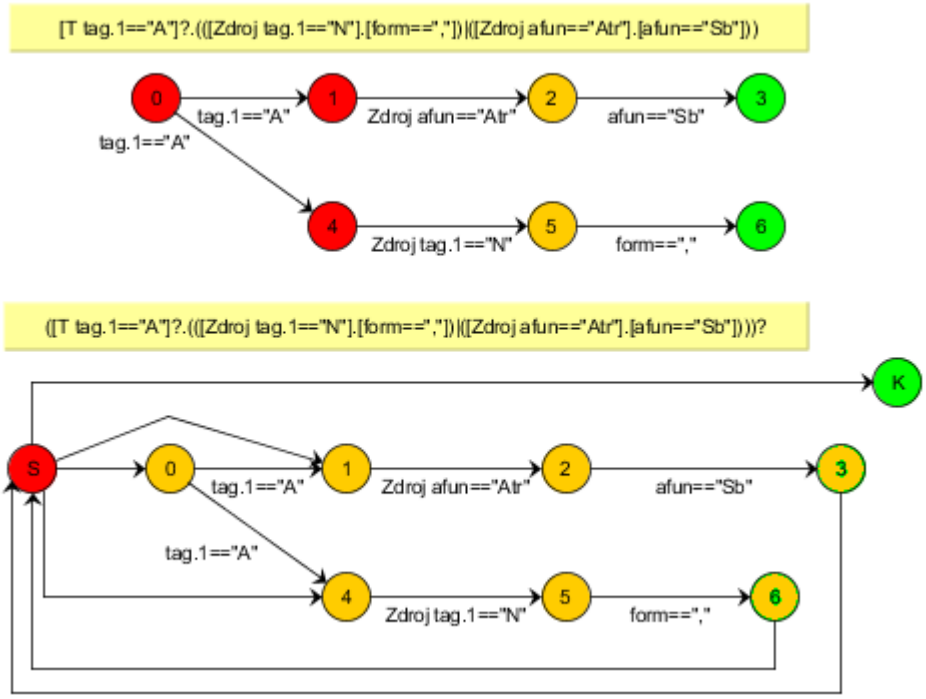
$$A.B = [\Sigma \cup \{\varepsilon\}, K_1 \cup K_2, Q_1, \delta, F_2]$$

Kde pre $k \in F_1, \delta_3(k, \varepsilon) = Q_2$ inak $\delta_3(l, a) = \emptyset$ *a potom*

$$\forall k \in K_1 \cup K_2, \forall a \in \Sigma \cup \{\varepsilon\}, \delta(k, a) = \delta_1(k, a) \cup \delta_2(k, a) \cup \delta_3(k, a)$$

- "*"-Operátor hviezdička je unárni operátor, ktorý automat upravý tak, že pridá nový iniciálny stav S, z ktorého vedú ε hrany do všetkých predchádzajúcich iniciálnych stavov. Vytvorí nový koncový stav K, do ktorého bude viesť jediná ε hrana z počiatočného stavu S. No a nakoniec zo všetkých koncových stavov sa zavedie ε hrana do počiatočného stavu S.

Príklad je na obrázku 8.



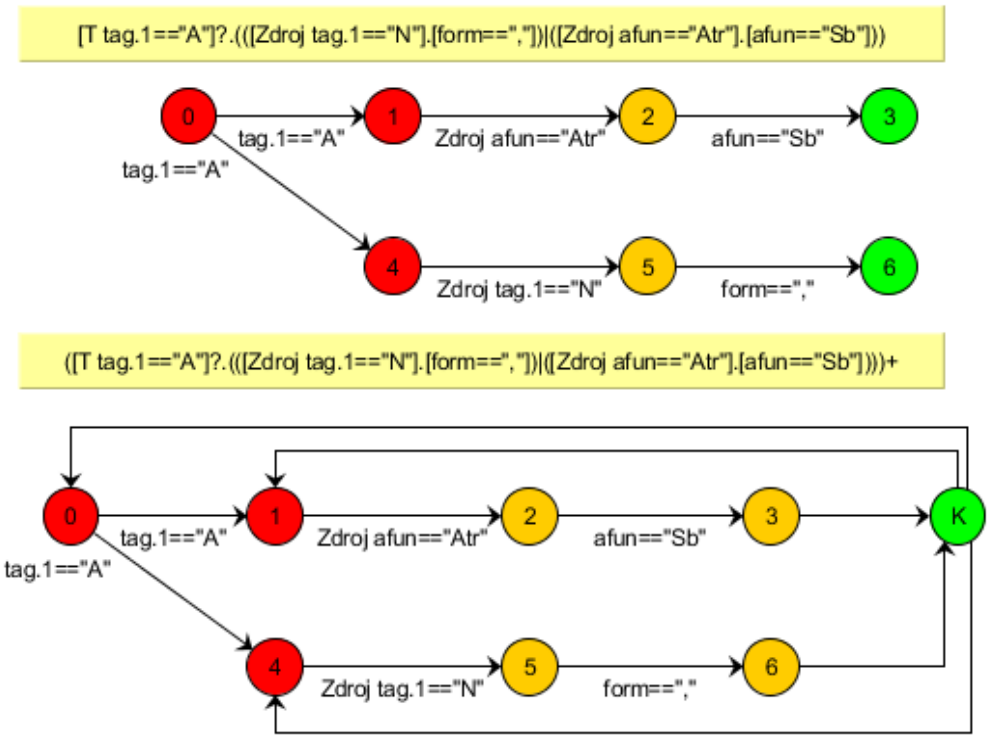
Obrázok 9 Príklad použitia operátora * na automat

Formálnejí: $A = (\Sigma, K_1, Q_1, \delta_1, F_1)\ \ A^* = (\Sigma \cup \{\varepsilon\}, K_1 \cup \{S, K\}, \{S\}, \delta, \{K\})$

Kde $\delta_2(S, \varepsilon) = Q_1 \cup \{K\}, pre\ s \in F_1, \delta_2(s, \varepsilon) = \{S\}\ inak\ \ \delta_2(l, a) = \emptyset$

$$\forall k \in \mathrm{K}_1, \forall a \in \Sigma \cup \{\varepsilon\}, \delta(k, a) = \delta_1(k, a) \cup \delta_2(k, a)$$

- "+"- Operátor +, Podobný s operátorom *, Pridá sa koncový vrchol K, s ktorého pôjdu ε hrany do počiatočných stavov a do ktorého pôjdu ε hrany z koncových stavov. Príklad je vidieť na nasledujúcom obrázku 9.



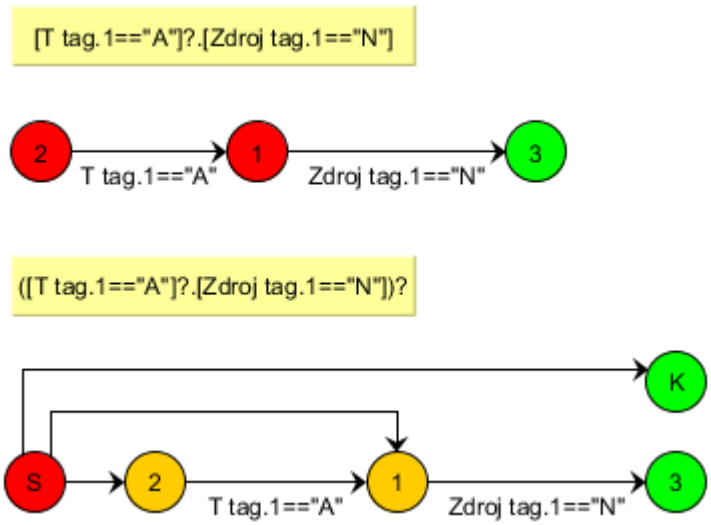
Obrázok 10 Príklad použitia operátora + na automat

Formálnejí: $A = (\Sigma, K_1, Q_1, \delta_1, F_1)\ A^+ = (\Sigma \cup \{\varepsilon\}, K_1 \cup \{K\}, Q_1, \delta, \{K\})$

Kde $\delta_2(K, \varepsilon) = Q_1, pre\ s \in F_1, \delta_2(s, \varepsilon) = \{K\}\ inak\ \ \delta_2(l, a) = \emptyset$

$\forall k \in \mathrm{K}_1, \forall a \in \Sigma \cup \{\varepsilon\}, \delta(k, a) = \delta_1(k, a) \cup \delta_2(k, a)$

- "?"- Operátor otáznik je unárny operátor, k pôvodnému automatu sa pridá nové stavy, začiatočný stav S, ktorý sa stane jediným začiatočným stavom. Z neho vedú ε hrany do všetkých predošlích počiatočných stavov. Pridá sa aj koncový stav K, do ktorého vedie jediná prázdna hrana z S. Príklad je na nasledujúcom obrázku 10.



Obrázok 11 Príklad použitia operátora ? na automat

Formálnejí:

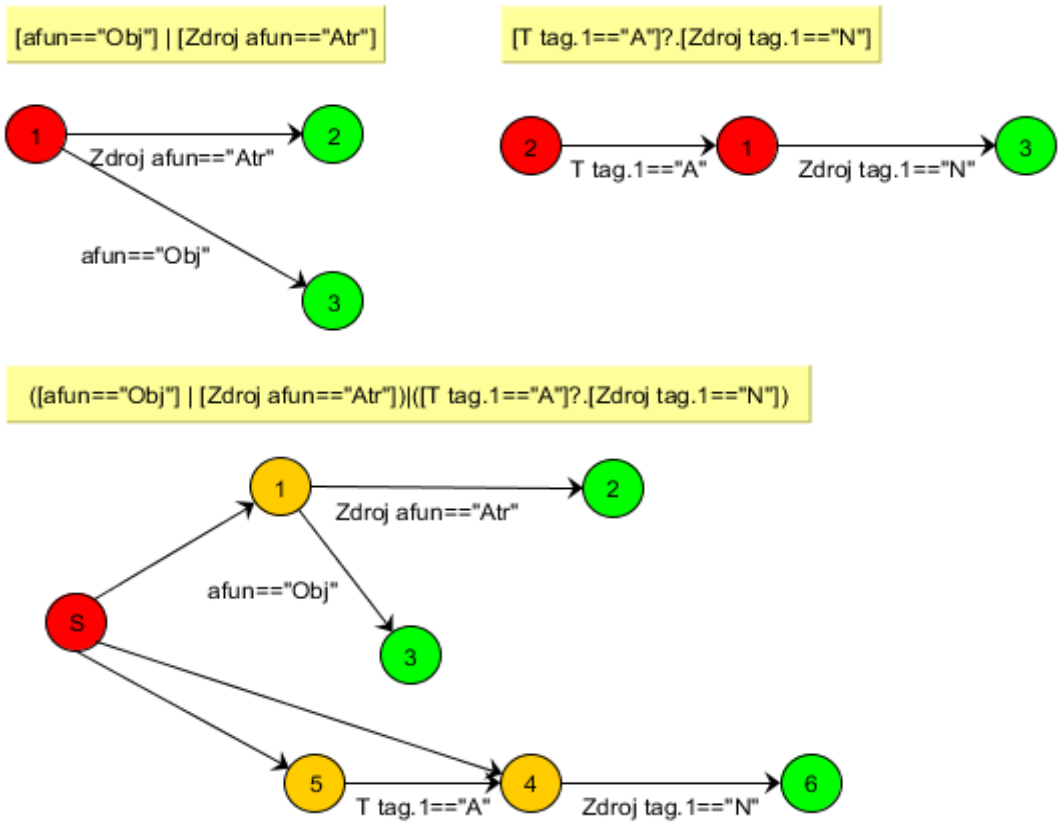
$$A = (\Sigma, K_1, Q_1, \delta_1, F_1\ )\ A^? = (\Sigma \cup \{\varepsilon\}, K_1 \cup \{S, K\}, \{S\}, \delta, F \cup \{K\})$$

$$\text{Kde } \delta_2(S, \varepsilon) = Q_1 \cup \{K\}, inak\ \delta_2(l, a) = \emptyset$$

$$\forall k \in \mathrm{K}_1, \forall a \in \Sigma \cup \{\varepsilon\}, \delta(k, a) = \delta_1(k, a) \cup \delta_2(k, a)$$

- "|"-Operátor | ako binárny operátor spojí 2 automaty tak, že pridá nový počiatočný vrchol S z ktorého vedú ε hrany do iniciálnych stavov oboch

automatov. S ostane jediným počiatočným stavom.



Obrázok 12 Príklad použitia operátora | na dva automaty

Formálnejí: $A = (\Sigma, K_1, Q_1, \delta_1, F_1), B = (\Sigma, K_2, Q_2, \delta_2, F_2)$

$$A|B = [\Sigma \cup \{\varepsilon\}, K_1 \cup K_2 \cup \{S\}, \{S\}, \delta, F_2 \cup F_1]$$

Kde $\delta_3(S, \varepsilon) = Q_2 \cup Q_1$ inak $\delta_3(l, a) = \emptyset$ *a potom*

$$\forall k \in K_1 \cup K_2, \forall a \in \Sigma \cup \{\varepsilon\}, \delta(k, a) = \delta_1(k, a) \cup \delta_2(k, a) \cup \delta_3(k, a)$$

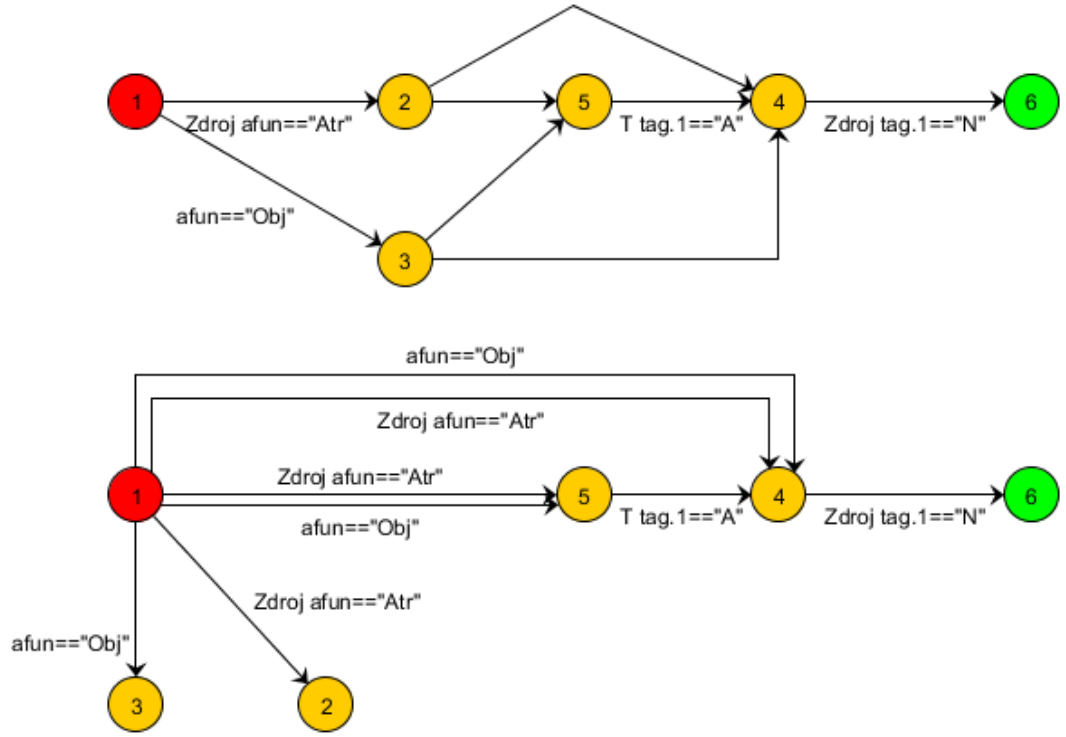
2.2.4 Odstránenie ε hrán z automatu

Po zložení automatu z výrazu je potrebné z neho odstrániť ε hrany. Postup je zhrnutý nasledujúcim algoritmom v Texte 9.

```
private void OdstranEpsilon(){
Stavy Automatu Stavy
(Pre každý stav S v Stavy)
{
   PrichádzajuceNeEpsilonHrany= PrichadzajuceNonEpsilonHrany(S);
   EpsilonUzaver= EpsilonUzaver(S);
   (Pre každú hranu H v PrichádzajúceNeEpsilonHrany)
   {
    (Pre každý stav K v EpsilonUzaver)
    {
     PridajHranu(H.ZdrojovyVrchol,K,H.Podmienka)
     if(S je iniciálny stav)
     {
      PridajStavMedziInicialneStavy(K)
     }
    }
   }
}
OdstranVsetkyEpsilonHrany();
OdstranZbytocneUzly();
```

Text 8 Pseudokód algoritmu na odstránenie ε hrán

Metóda EpsilonUzaver(S) nájde všetky stavy automatu, do ktorých je možné sa dostať zo stavu S len po ε hranách. Tiež predpokladám, že Hrana v tomto postupe obsahuje údaje o zdrojovom vrchole, cieľovom vrchole a podmienku vlastnosti slova. Príklad automatu po odstránení prázdnych ε hrán je na obrázku 12.



Obrázok 13 Príklad odstránenia ε hrán z automatu

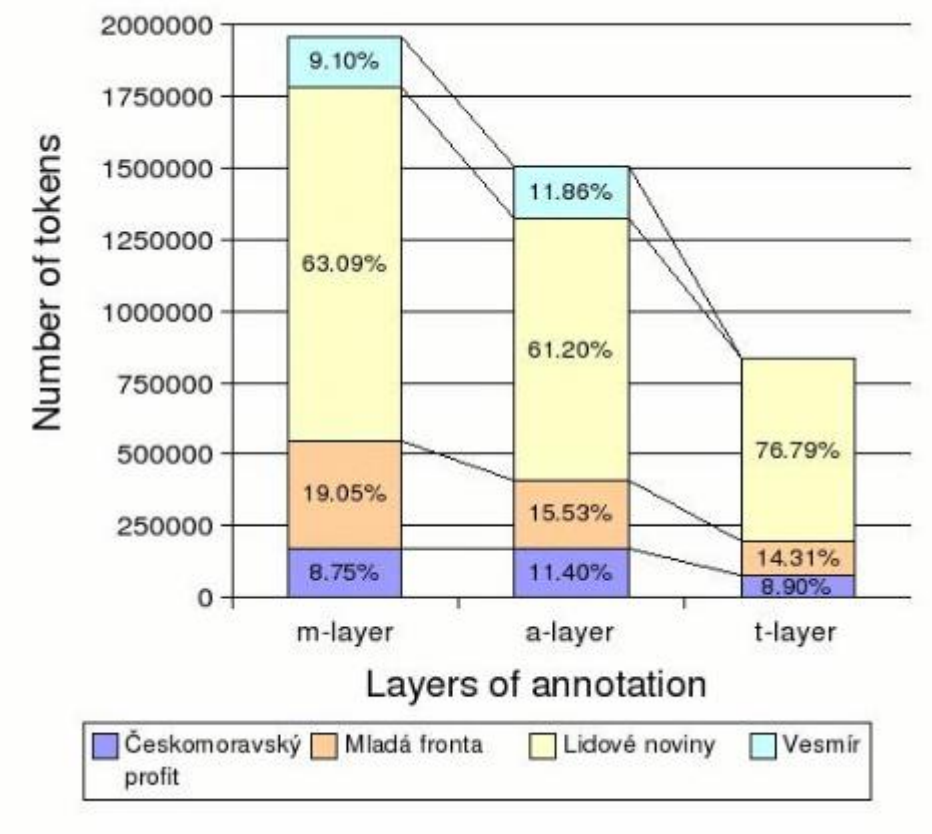
3.Evaluácia

3.1 Testovacie dáta

Testovacie dáta boli 2 druhy. Jeden z nich boli texty z PDT (pozri 1.2). Obsahovo sa jedná o články z nasledujúcich periodík[9]:

- Lidové noviny (denník), ISSN 1213-1385, 1991, 1994, 1995
- Mladá fronta Dnes (denník), 1992
- Českomoravský Profit (denník), 1994
- Vesmír (vedecký časopis), ISSN 1214-4029, Vesmír, s.r.o., 1992, 1993

Rozloženie slov celého korpusu z jednotlivých časopisov vyjadruje nasledujúci graf[9]



Obrázok 14 Rozloženie dát v PDT2.0

Number of tokens (počet tokenov) vyjadruje počet uzlov na jednotlivých rovinách anotácie (pozri 1.2.1). Na evaluáciu bola použitá časť dát, konkrétne train-1 z CD-ROM obsahujúcim PDT, s rozsahom 43790 slov.

Druhý zdroj dát na evaluáciu je anotovaná knižka Arthur Conan Doyle *Studie v šarlatové*, ktorý na rozdiel od dát s PDT má automaticky anotovanú M-rovinu a A-rovinu. Rozsah diela je 43266 slov. Koreferencie v obidvoch zdrojoch sú však ručne anotované na M-rovine.

3.2 Spôsob evaluácie

Evaluácia pravidiel sa robila spôsobom v ktorom každé slovo patriace do koreferenčného páru predikovaného pravidlom a zároveň patrilo nejakému anotovanému koref. páru je označené za úspech aj keď druhá časť páru je anotovaná inak ako predikovaná. Ak máme slová A,B,C,D. Kde anotovaný pár je A-C, pravidlo predikovalo pár A-B. Potom

- A – je úspešne označené slovo (true positive TP)
- B – je neúspešne označené slovo (false positive FP)
- C – je neúspešne neoznačené slovo (false negative FN)
- D – je úspešne neoznačené slovo (true negative TN)

Samozrejme za predpokladu, že B,D nie sú súčasťou nejakého iného koreferenčného páru. Štatistické údaje, ktoré sa vypočítavajú sú **precision**, **recall**, **F-measure**. Kde vzorce pre jednotlivé údaje sú nasledujúce.

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Tento spôsob evaluácie je používaný, napríklad aj v práci[17] kde hodnota F-measure figuruje ako F-chains.

3.3 Kategórie pravidiel a ich vyhodnotenie

TODO

4.Aplikácia Koreferencie

4.1 TrEd

TrEd[16] je plne nastaviteľný a programovateľný grafický editor a prehliadač stromových štruktúr, ako napríklad závislostné stromy. Medzi iným, bol použitý ako hlavný nástroj na anotáciu syntaktickej a tectogramatickej roviny pre texty z PDT. Program Koreferencie využíva negrafickú variantu TrEd - btred, užitočnú pre dávkové spracovanie dát. Viac o spojení s btredom je v 4.3. TrEd je flexibilný nástroj je stále vo vývoji a bolo vypracovaných mnoho zásuvných modulov, ktoré rozširujú jeho funkcionalitu. Modul Play the language je potrebný na čítanie koreferencií na M-rovine, preto je pre používanie aplikácie je kľúčový.

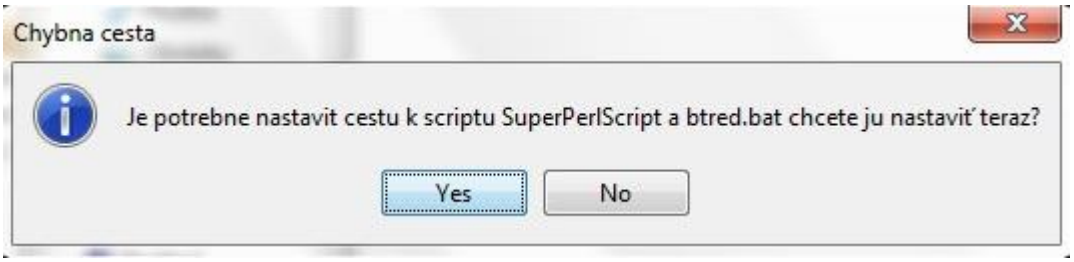
4.2Užívateľská dokumentácia

4.2.1 Inštalácia

Nutnou podmienkou k používaniu programu Koreferencie je mať nainštalovaný program TrEd s nainštalovaným pluginom Play the Language. Program tiež potrebuje nainštalovanú Java Virtual Machine. Inak program je priamo spustiteľný súbor s príponou .jar vo všetkých operačných systémoch. Ak nemáte nastavené súbory .jar ako priamo spustiteľné, je možné spustiť z príkazovej riadky pomocou príkazu java –jar Koreferencie.jar.

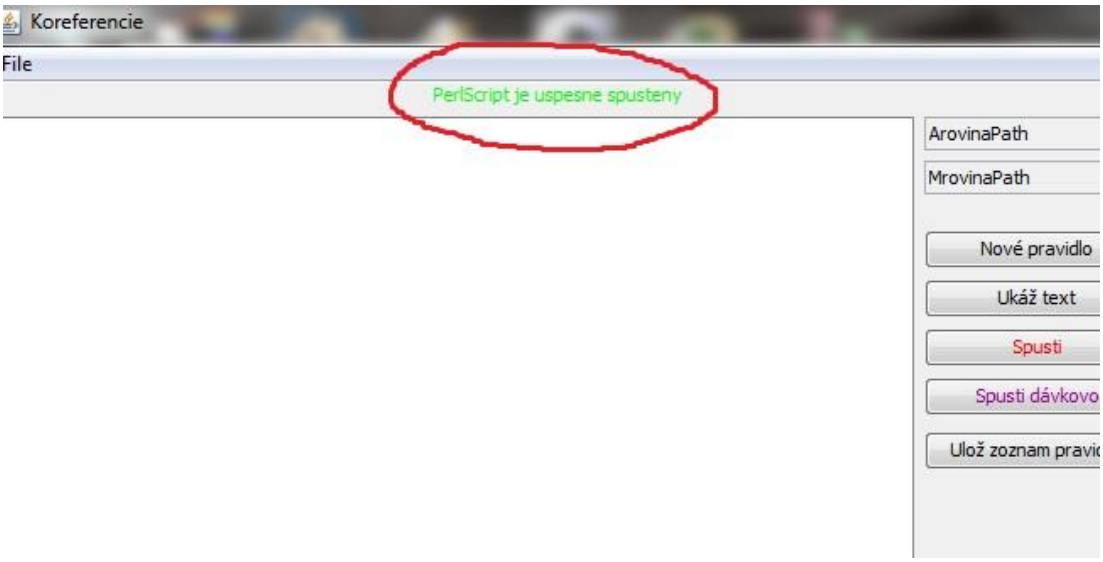
4.2.2 Spustenie

Po prvom spustení vyžaduje program vyhľadať cestu k Tredu, konkrétne cestu k jeho podprogramu btred. V operačnom systéme Windows je to súbor btred.bat v zložke kde je nainštalovaný tred. V operečnom systéme Linux, je buď potrebné nájsť súbor „btred“ alebo „start_btred“. Podľa toho akú verziu Tredu máte nainštalovanú. Toto je nutný predpoklad k tomu aby sa vytvorilo spojenie s Tredom a bolo možné pracovať s textom.



Obrázok 15 Dialógové okno pri prvom spustení programu

Cesta k súboru SuperPerlScript.pl nie je potrebné nastaviť ak sa štandardne nachádza v zložke s programom. Ak sa spojenie s Tredom vytvorilo úspešne, v základnom okne bude napísané "PerlScript je úspešne spustený" ako vidieť na nasledujúcom obrázku.



Obrázok 16 Perlscript je úspešne spustený

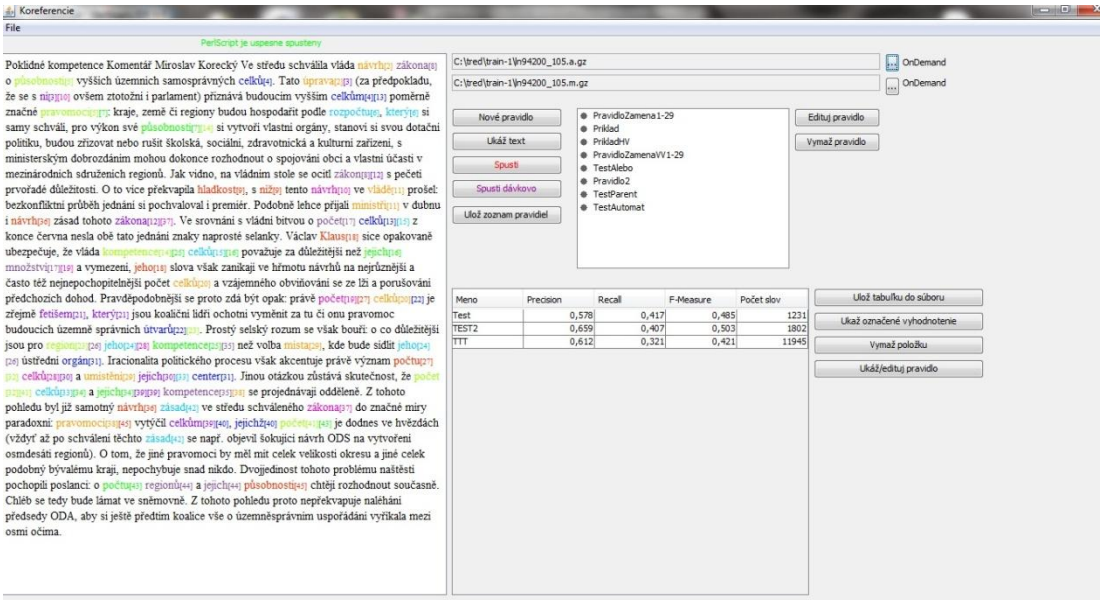
4.2.3 Formát súborov s textom

Vstupné dáta sú v pml formáte. O PML formáte je podrobne na stránkach [5].
V programe Koreferencie sa používajú prvé tri roviny, W-rovina, M-rovina, A-rovina. Pre každú rovinu je osobitne jeden súbor s koncovkami „w.gz", „m.gz" a „a.gz" pre W,M,A-rovinu v tomto poradí. Súbory jedného textu by mali byť spolu v jednej zložke. Automaticky anotovať text je možné pomocou nástroja tool_chain, podrobnejší postup je na adrese [19]. Aby bolo možne urobiť evaluáciu pravidiel, musí byť M-rovina navyše s anotovanými koreferenčnými pármi. Príklad anotácie koreferencie na M-rovine je v Texte 9.

```
<m>
  <AM id="m-ln94206-147-p2s1Bw22">
   <src.rf>manual</src.rf>
   <w.rf>w#w-ln94206-147-p2s1Bw22</w.rf>
   <form>Budějovice</form>
   <lemma>Budějovice_;G</lemma>
   <tag>NNFP1-----A----</tag>
   <coref>
    <target-node.rf>m-ln94206-147-p2s1Aw2</target-node.rf>
    <type>textual</type>
   </coref>
  </AM>
 </m>
```

Text 9 Anotácia koreferencie na M-rovine

4.2.4 Základná obrazovka



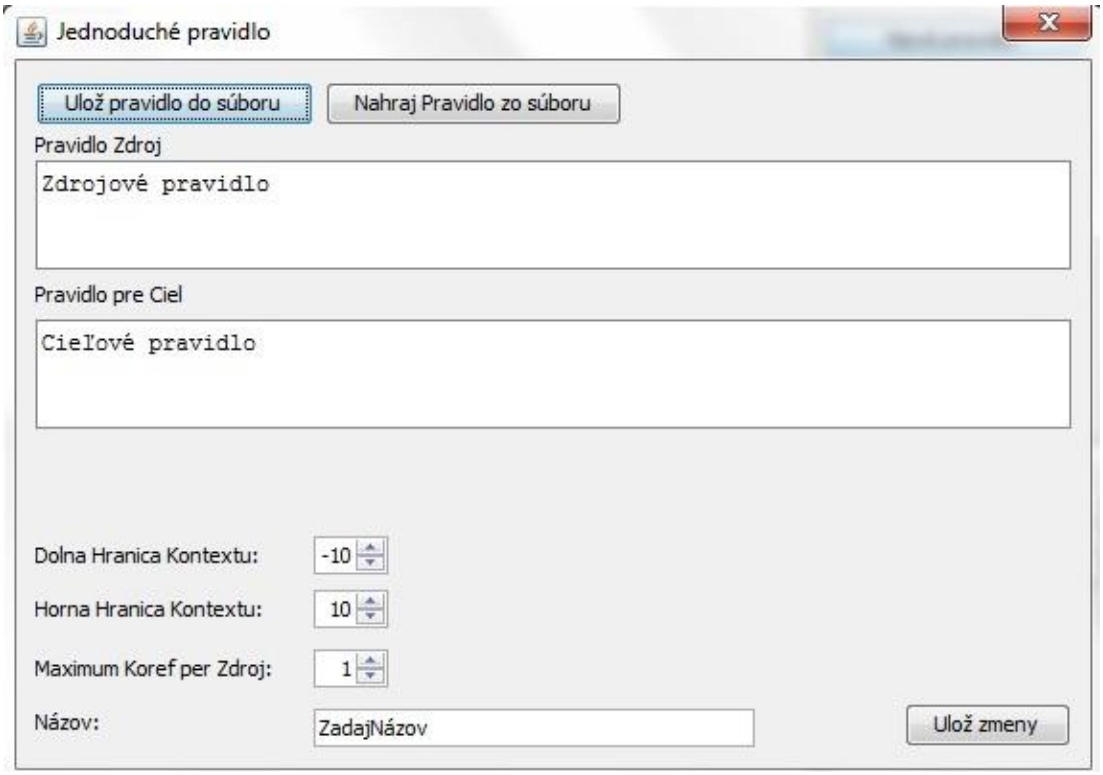
Obrázok 17 Základná obrazovka

Na obrázku vidieť základnú obrazovku programu Koreferencie. Veľká plocha vľavo je určená na vizualizáciu textu s vyznačenými koreferenciami. Koreferencie sú jednak farebne vyznačené a okrem toho koreferenčné páry majú spoločný dolný pravý index. Niektoré slová koreferujú viac ako s jedným ďalším slovom majú viac indexov. Vpravo sú postupne zhora nadol nasledujúce možnosti.

- Načítať súbor A roviny, ktorého text sa automaticky zobrazí na ľavej ploche. Tlačidlom „...“ sa otvorí dialógové okno pre výber súboru. Súbory sú s príponou „.a.gz“.
- Načítať súbor M roviny, načíta sa automaticky keď sa načíta súbor A roviny a je v tej istej zložke. Tlačidlom „...“ sa otvorí dialógové okno pre výber súboru.
- Zoznam pravidiel. Tu sú jednotlivé vy tvorené pravidla a možnosť práce s nimi. Podrobný popis jednotlivých možností bude nasledovať
- Tabuľka vyhodnotení. Sem sa ukladajú vyhodnotenia pravidiel na text. Tiež je tu možnosť si ich prezerať.

4.2.5 Vytváranie a editácia pravidiel

Vytvoriť nové pravidlo je možné so základnej obrazovky tlačidlom „Nové pravidlo“. Otvorí sa nasledujúce dialógové okno. V základnej obrazovke je možné po označení ešte pravidlo editovať alebo vymazať tlačidlami „Edituj pravidlo“, „Vymaž pravidlo“. Celý zoznam sa uloží pre ďalšie spustenie tlačidlom „Ulož zoznam pravidiel“.



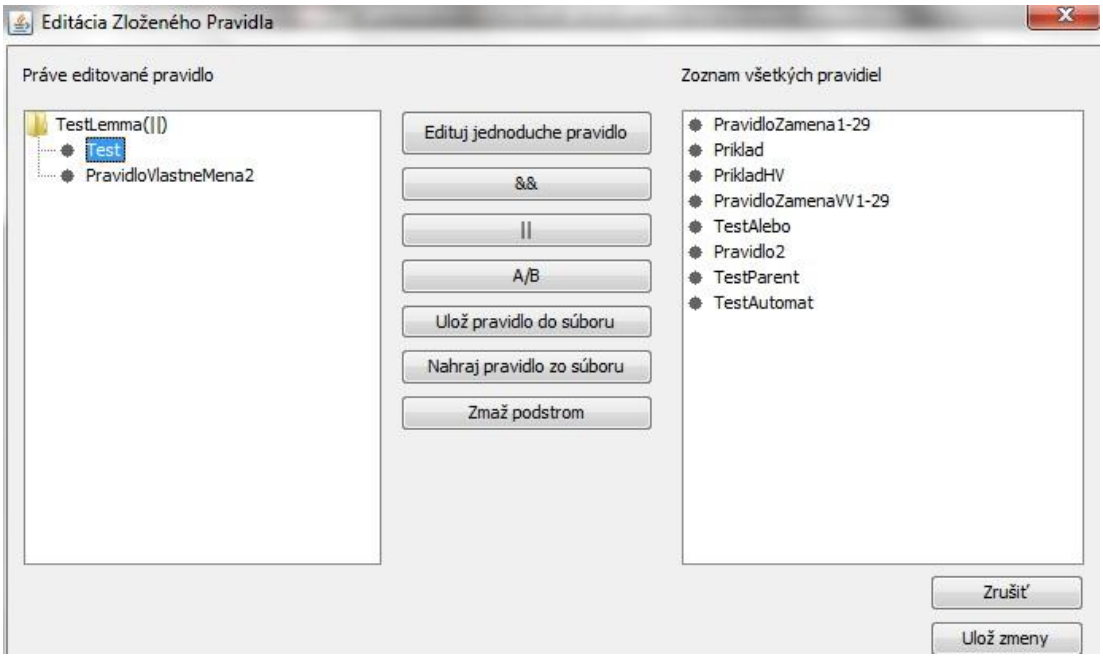
Obrázok 18 Dialógové okno jednoduchého pravidla

Zhora nadol sú možnosti

- Uložiť/Nahrať pravidlo do súboru. Tvar súboru pre jednoduché pravidlo je textový súbor v ktorom prvý riadok pravidlo pre Zdroj, druhý riadok pravidlo pre Ciel a riadky 3-6 obsahujú v poradí: dolnú hranicu kontextu, hornú hranicu kontextu, Maximálny počet koreferencií per Zdroj, Názov pravidla.
- Pravidlo pre Zdroj. Sem sa napíše pravidlo pre hľadanie Zdroja, teda prvého člena z koreferenčnej dvojice. Musí obsahovať premennú Zdroj. Pozri kapitola 2.1.2 pre syntax a význam pravidiel.
- Pravidlo pre Ciel. Sem sa napíše pravidlo pre hľadanie Ciela, teda druhého člena z koreferenčného páru. Pre každý Zdroj sa v jeho okolí hľadajú Ciele. Pravidlo musí obsahovať premennú Ciel, syntax pravidiel je v kapitole 2.1.2.
- Dolná hranica kontextu je nekladné celé číslo, udáva koľko slov naľavo od Zdroja sa má prehľadávať jeho okolie pri hľadaní Ciela.
- Horná hranica kontextu je nezáporné celé číslo, udáva koľko slov napravo od Zdroja má byť prehľadávané jeho okolie.

- Maximum Koref per Zdroj je položka pravidla stanovujúca maximum koľko Cielov sa má nájsť pre jeden Zdroj. Priorita pri hľadaní je nastavená tak, že sa berú prvé Ciele naľavo od Zdroja a potom prvé Ciele napravo.
- Názov, sem sa udáva názov pravidla.
- Tlačidlom „Ulož zmeny“ sa uložia zmeny a dialógové okno sa zatvorí. Ak zmeny nechcem uložiť stačí zatvoriť okno štandardne.

Po skončení práce v dialógovom okne Jednoduché pravidlo, je možné spojiť toto pravidlo s ďalšími pravidlami a vytvárať tak zložené pravidlo v dialógovom okne „Editácia Zloženého Pravidla“.



Obrázok 19 Dialógové okno Editácia zloženého pravidla

V tomto okne je naľavo práve editované pravidlo. Ak je to zložené pravidlo, má podobu stromu, kde vnútorné uzly sú množinové operátory a listy sú jednoduché pravidlá. Úplne napravo je zoznam všetkých pravidiel, je to kópia zoznamu zo základnej obrazovky. V strede sú zhora nadol nasledujúce možnosti.

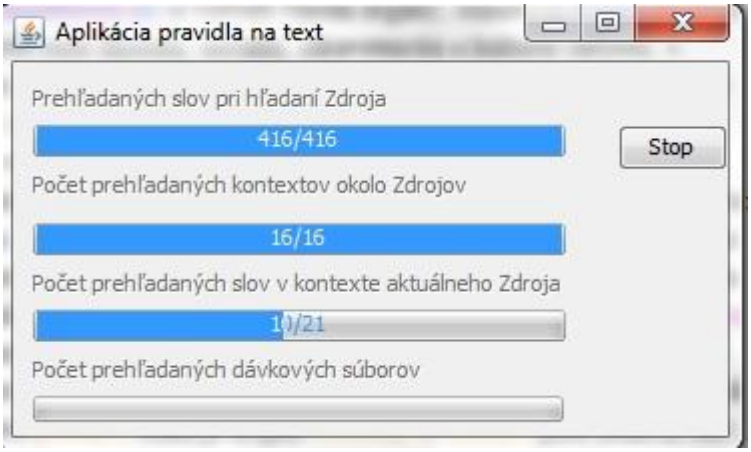
- Edituj jednoduché pravidlo, pri tejto možnosti je potrebné mať označené jednoduché pravidlo z ľavého pravidla. Otvorí sa dialógové okno „Jednoduché pravidlo“.
- &&,||,A/B- Množinové binárne operátory, ich význam je popísaný v kapitole 2.1.3. Je teda potrebné mať označený uzol práve editovaného pravidla a niektoré pravidlo zo zoznamu všetkých pravidiel napravo. Ďalšou možnosťou je mať označený len niektorý uzol práve editovaného pravidla, v tom prípade sa otvorí dialógové okno „Jednoduché pravidlo“. V ktorom je možné vytvoriť nové jednoduché pravidlo (alebo nahrať nejaké zo súboru), ktoré sa použije namiesto vyznačeného pravidla z pravého zoznamu.
- Ulož / Nahraj pravidlo do/zo súboru. Pravidlá sa ukladajú do súboru s príponou .rule. Existujú teda 2 druhy súborov pre pravidlá, jednoduché pravidlá je možné ukladať do textového súboru v dialógovom okne „Jednoduché pravidlo“, obecné pravidlo je možné uložiť do súboru s príponou .rule.
- Zmaž podstrom. Ak chceme zmazať nejakú časť editovaného pravidla, je to možné označením uzlu v ľavom zozname a stlačením tohto tlačidla sa odstráni uzol aj so všetkými potomkami. Nadradený operátor sa nahradí bratom označeného uzlu.

Ak chceme uložiť zmeny v pravidle vhodné tlačidlo „Ulož zmeny“, ktoré tiež uzavrie dialógové okno. Ak zmeny uložiť nechceme, stačí okno zavrieť klasicky, alebo tlačidlom „Zrušiť“.

4.2.6 Aplikácia pravidla

Aplikovať pravidlo je možné 2 spôsobmi. Prvým spôsobom je aplikovať na jeden súbor s textom. Predtým je potrebné mať vybratý súbor A-roviny, pomocou tlačidla „...“ v hornej pravej časti základnej obrazovky, pozri 4.2.4. Tiež je potrebné mať označené pravidlo v zozname pravidiel. Potom tlačidlom „Spusti“ začne proces aplikácie pravidla, ukáže sa okno ukazujúce stav aplikácie pravidla. Tak ako na nasledujúcom obrázku. Proces je možné zastaviť tlačidlom Stop. Tiež sa môže stať že program vypíše nejakú syntaktickú chybu pri aplikácií pravidla. Napríklad

chýbajúca zátvorka alebo neznáme kľúčové slovo. Nájdené dvojice sa vizualizujú v ľavej časti základnej obrazovky.

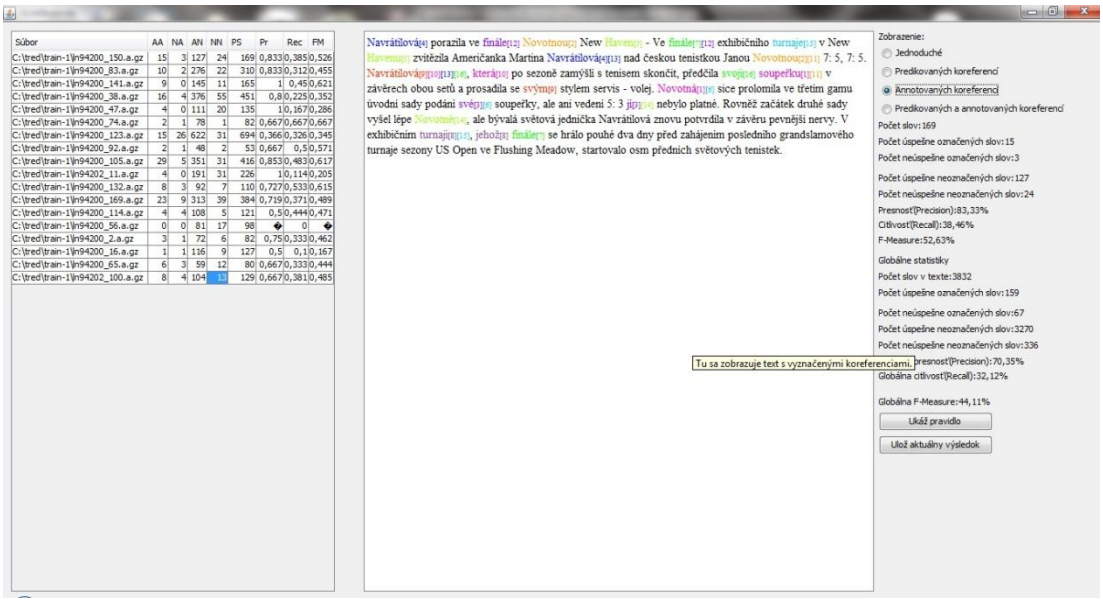


Obrázok 20 Priebeh práce pri aplikácií pravidla na text

Druhým spôsobom aplikovania je tlačidlom „Spusti dávkovo“. Týmto spôsobom sa otvorí dialógové okno v ktorom je možné vybrať viac súborov A roviny, na ktoré sa bude pravidlo aplikovať. Klávesnicou je možné vybrať súbory pomocou SHIFT+šípky. Po vybraní súborov sa otvorí okno z obrázku 20 s tým rozdielom, že posledná položka „Počet prehľadaných dávkových súborov“ bude aktívna. Po aplikovaní pravidla sa otvorí okno s výslednou evaluáciou o ktorej je viac v kapitole 4.2.7

4.2.7 Evaluácia

Po aplikovaní pravidla dávkovo na viac súborov, alebo po vybraní výsledku predošlých aplikácií pravidiel na text sa otvorí okno s evaluáciou.



Obrázok 21 Okno evaluácie

Okno má 3 časti. Úplne vľavo je tabuľka kde jednotlivé položky sú súbory na ktoré sa pravidlo aplikovalo. Údaje v tabuľke sú v poradí

- Cesta k súboru
- AA- Počet úspešne nájdených slov v texte súboru
- NA- Počet neúspešne označených slov v texte súboru
- AN- Počet úspešne neoznačených slov v texte súboru
- NN- Počet neúspešne neoznačených slov v texte súboru
- PS- Počet slov v texte súboru.
- Pr- Precision, štatistický údaj presnosti. Hodnota je v percentách/100
- Rec- Recall, štatistický údaj pokrytia. Hodnota je v percentách/100
- FM- F-Measure, štatistický údaj, vyjadruje celkovú úspešnosť pravidla. Hodnota je v percentách/100

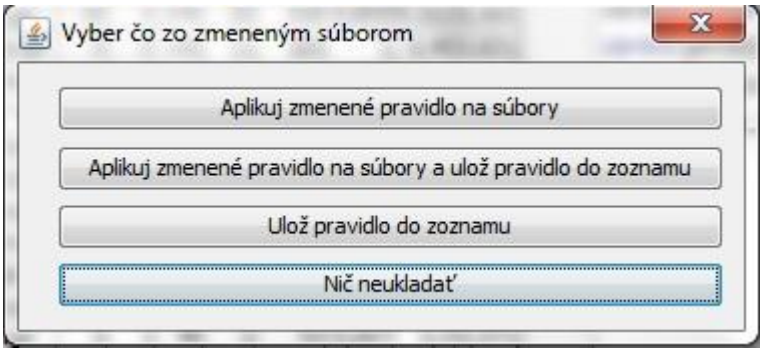
Dvojklikom na jednotlivé položky sa text objaví v strednej časti okna a štatistické údaje v pravej časti okna. V pravej hornej časti je možne vybrať 4 druhy vizualizácie textu.

- Jednoduché- Tento typ vizualizácie zobrazuje text bez vyznačených koreferencií. Ukazuje však zelenou farbou slová ktoré boli predikované pravidlom a zároveň anotované. Červenou farbou sú slová anotované v texte, ktoré pravidlo nepredikovalo. Modrou sú slová, ktoré boli predikované pravidlom, ale neboli anotované v texte.
- Predikovaných koreferencií- Tento typ zobrazenia ukáže text s predikovanými koreferenciami, teda tými ktoré označilo práve použité pravidlo
- Annotovaných koreferencií – Zobrazenie vizualizuje text s anotovanými koreferenciami.
- Predikovaných aj annotovaných koreferencií – Zobrazenie kombinujúce všetky predchádzajú prístupy. Zobrazí slová vo farebnej schéme ako v jednoduchom zobrazení. Zároveň pravé dolné indexy označujú anotované koreferencie a ľavé dolné indexy označujú koreferencie predikované pravidlom.

Pod možnosťami zobrazenia sú zaradom štatistické údaje vyhodnotenia textu. Za ním nasledujú štatistické údaje globálne, teda údaje zo všetkých súborov textu dokopy.

Možnosťami, čo ďalej je tlačidlo „Ukáž pravidlo“. Ukáže sa dialógové okno „Editácia Zloženého pravidla“, v ktorom je možné si prezrieť použité pravidlo. Ak v tomto okne dáme „Uložiť zmeny“, ponúknu sa nasledujúce možnosti.

- Aplikuj zmenené pravidlo na súbory – Aplikuje pravidlo na tie isté súbory na aké sa vzťahovala evaluácia. Stratia sa tým výsledky súčasnej evaluácie.
- Ulož pravidlo do zoznamu – Uloží pravidlo do zoznamu pravidiel
- Aplikuj zmenené pravidlo na súbory a ulož pravidlo do zoznamu- kombinácia dvoch vyššie uvedených.
- Nič neukladať – žiadna akcia, späť do okna evaluácie.



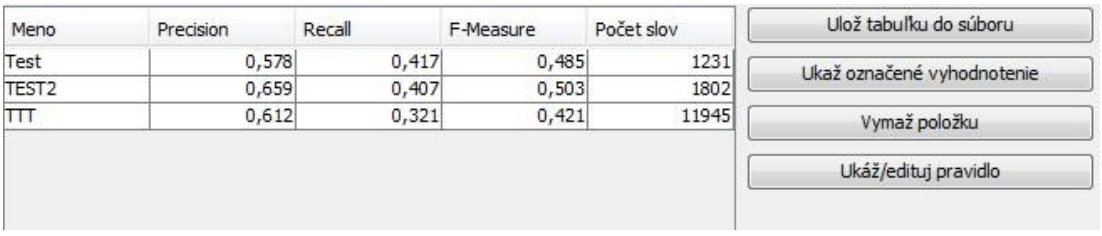
Obrázok 22 Možnosti po editovaní pravidla v evaluácií

Ďalšou možnosťou v okne evaluácie je tlačidlo „Ulož aktuálny výsledok“. Týmto sa uložia dáta o vyhodnotení do tabuľky, ktorá je v základnej obrazovke. Poslednou možnosťou je klasicky zavrieť okno, vtedy sa nič neuloží a program sa vráti do základnej obrazovky.

4.2.8 Práca s tabuľkou s výsledkami evaluácie

V základnej obrazovke sa nachádza tabuľka, kde sú uložené jednotlivé výsledky aplikovania pravidiel na text. Okrem názvu sú v tabuľke globálne štatistické údaje aplikácie pravidla. Možnosti práce s tabuľkou sú nasledujúce

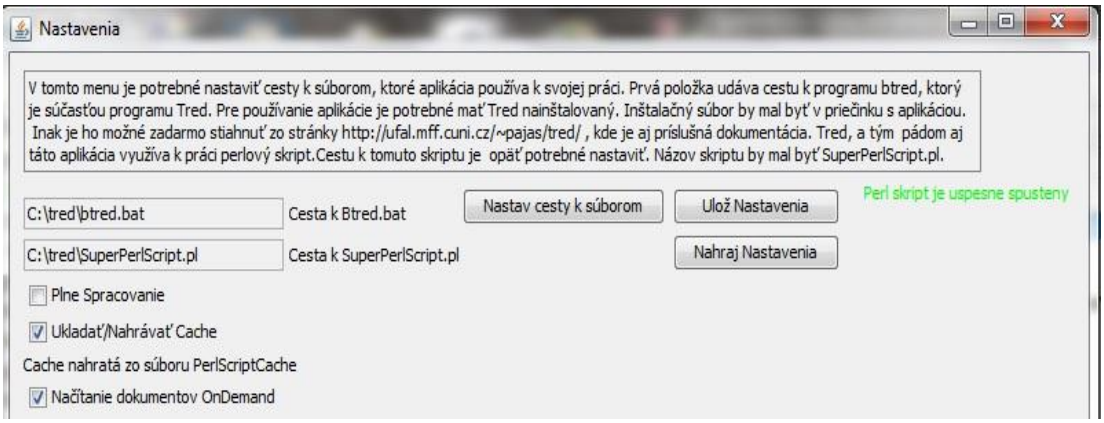
- „Ulož tabuľku do súboru“ – Uloží celú tabuľku do súboru, ktorý sa znova pri spustení aplikácie nahrá.
- „Ukáž označené vyhodnotenie“ – Spustí okno evaluácie, pozri 4.2.7.
- „Vymaž položku“
- „Ukáž/edituj pravidlo“ – Spustí dialógové okno „Editácia Zloženého pravidla“. Po uložení zmien, sú možnosti na obrázku.



Obrázok 23 Tabuľka s uloženými výsledkami evaluácie

4.2.9 Nastavenia

V základnej lište **File->Nastavenia** je prístupné okno s nastaveniami.



Obrázok 24 Okno s nastaveniami programu Koreferencie

Najdôležitejšie nastavenie cesty k programu btred a skriptu SuperPerlScript.pl. Zatiaľ čo SuperPerlScript.pl je súbor priložený k programu a mal by si cestu k nemu nájsť sám, je potrebné nastaviť cestu k programu btred. Bez spojenia s btred totiž nemôže program fungovať. Možnosti v okne sú nasledujúce.

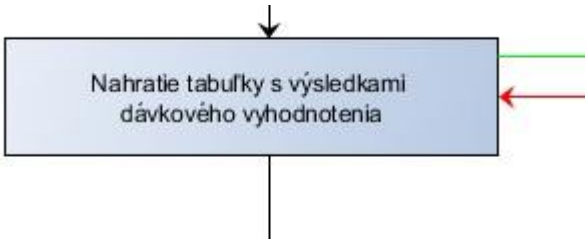
- „Nastav cesty k súborom“- Týmto tlačidlom sa nastavia cesty k programu btred a ak je na neštandardnom mieste k súboru SuperPerlScript.pl.
- „Ulož Nastavenia“ – Uloží nastavenia do súboru. Automaticky sa nahrajú pri ďalšom spustení programu.
- „Nahraj Nastavenia“ – Nahrá nastavenia zo súboru.
- „Plné Spracovanie“ – Ak nie je zaškrtnuté, pri aplikácií pravidla na viac súborov, ignoruje tie ktoré nemajú anotované žiadne koreferencie. Je to kvôli tomu aby takéto súbory neskresľovali celkovú evaluáciu. Ak chcete použiť pravidla aj na neanotované texty, zaškrtnite túto možnosť
- „Ukladať/Nahrávať cache“ – Program používa cache na pre dotazy na btred. Ak chcete aby sa po ukončení uložila táto cache do súboru, zaškrtnite túto možnosť. Veľkosť súboru závisí na množstve dát na ktorých ste pracovali. Pri štarte sa cache nahrá do pamäti.
- „Načítanie dokumentov OnDemand“ – Program ako bolo používa cache na pre dotazy na btred, ak je zapnutá funkcia OnDemand, program nedá Tredu načítať nový súbor dovtedy, kým sa neobjaví dotaz na ktorý nie je v cache odpoveď. Zrýchľuje aplikáciu pravidiel, odporúča sa nechať zapnuté.

4.3 Programátorská dokumentácia

4.3.1 Dátový model a graf toku programu

Podrobný popis dátového modelu a graf toku sa nachádza v prílohách B a C. V prílohe B je flowchart s nasledujúcimi prvkami. Podrobná dokumentácia zdrojového kódu vygenerovaná pomocou javadoc je v prílohe D.

- Proces alebo úkon, ktorý urobí program. Príklad na obrázku 23.



Obrázok 25 Proces v grafe toku v programe

- Proces alebo úkon, ktorý urobí užívateľ programu. Môže to byť aj stlačenie tlačidla, nastavenie hodnoty a podobne. Príklad je na obrázku 24.

Obrázok 26 Príklad procesu alebo úkonu uživatela

- Dátová štruktúra, súbor alebo iná informácia obsahujúca dáta. Pre každý dôležitý zdroj je podrobnejší popis v prílohe C.

Obrázok 27 Dátová štruktúra

- Prvok rozvetvujúci tok programu. Môže mať ľubovoľné množstvo výstupov, pri ktorých je popis rozhodnutia. Popis chýba ak je zrejmý.

Obrázok 28 Prvok rozhodovania v programe

- Ďalšie intuitívne poznateľné prvky ako začiatok programu, pevný disk apod.

Prvky programu sú v grafe spojené 3 druhmi šípok. Čierna šípka označuje smer toku programu. Každý proces by mal mať aspoň jeden vstup a práve jeden výstup po čiernej šípke. Červené šípky smerujú od dátových štruktúr k procesom a naznačujú že tento zdroj proces potrebuje. Zelené šípky smerujú od procesu k zdrojom a dátam, naznačujú výstupy daného procesu.

4.3.2 Prepojenie Tred s programom Koreferencie

Zaujímavým prvkom programu je prepojenie Tredu, konkrétne jeho časti pre dávkové spracovanie btred, s programom Koreferencie. Motiváciou spojenia bolo využiť schopnosť btredu čítať formát PML aj s anotáciou koreferencií na M-rovine. Bolo treba nájsť spôsob ako načítať súbory textu na W,M,A - rovinách a potom sa dotazovať zvnútra Javy na jednotlivé údaje. Najlepšie tak aby sa btred nemusel znovu spúšťať a vypínať po jednotlivých dotazoch alebo načítaní iného dokumentu. Btred má možnosť sa spustiť s perlovým skriptom ako parametrom pre svoje fungovanie.

Prvým problémom bolo vytvoriť obojstranné spojenie, tak aby to čo vypisuje btred na svojom štandardnom aj chybovom výstupe bolo na druhej strane v Jave prečítané. Ak totiž ostal výpis btredu neprečítaný, zaseklo sa spojenie. Taktiež bolo treba vyriešiť problém s čítaním českých znakov.

Pre čítanie chybové výstupu z btred slúži trieda StreamGobbler, ktorá v osobitnom vlákne číta všetko z chybového výstupu. Štandardný výstup je čítaný stále vtedy keď sa očakáva na ňom nejaký výstup, je to možné vďaka použitému perlovému skriptu, kde je štandardný výstup plne pod kontrolou. Takže bolo možné presne určiť kedy čítať údaje a kedy naopak vkladať údaje na štandardný vstup. České znaky je možné čítať pri nastavení kódovanie UTF8.

Najprv bolo vytvorených niekoľko druhov perlových skriptov, ktoré slúžili pre rôzne druhy dotazov a nedokázali udržať spojenie s btredom po zmenení dokumentu. To vyvolávalo značné spomalenie celého programu. Celé sa to značne zrýchlilo zjednotením všetkých skiptov do jedného, s názvom SuperPerlScript.pl, ktorý je priložený k programu. Dokáže fungovať s btredom aj samostatne, spúšťa sa príkazom **btred -I SuperPerlScript.pl --init SPS::poslouchej().** Následne je spustené prostredie príkazovej riadky, v ktorej je možné dávať dotazy.

Počet dotazov na btred sa stal kritickým pre rýchlosť aplikovania pravidla. Riešením teda bolo vytvoriť cache kde ukladať už položené dotazy. Týmto spôsobom sa rapídne zrýchlilo aplikovanie pravidiel, najmä opakované aplikácie toho istého pravidla, alebo pravidla využívajúce tie isté údaje. Pri dávkovom spracovaní však stále bolo potrebné čakať kým sa načítajú jednotlivé dokumenty v btred aj vtedy ak následne všetky potrebné údaje pre dotazy už boli uložené v cache. Kvôli tomuto bola implementovaná funkcia OnDemand, ktorá keď je zapnutá (pozri 4.2.9) načítava dokumenty práve vtedy keď príde dotaz na ktorý nie je odpoveď v cache.

Spojenie programu s btred obstaráva trieda PerlScript.

Literatúra

[1] M. Chytil: Automaty a gramatiky, SNTL, Praha 1984

[2] Jasoň: Automaty a gramatiky,1996

[3] http://sk.wikipedia.org/wiki/Kone%C4%8Dn%C3%BD_automat

[4] http://montcs.bloomu.edu/~bobmon/Information/RPN/infix2rpn.shtml

[5] http://ufal.mff.cuni.cz/jazz/PML/doc/pml_doc.html

[6] http://ufal.mff.cuni.cz/~hladka/project/2008-09/PFL054_2008_09_project.pdf

[7] Nguy Giang Linh, Zdeněk Žabokrtský, Rule-based Approach to Pronominal Anaphora Resolution, Applied on the Prague Dependency Treebank 2.0 Data,

[8] Dizertační práce, Anna Nedoluzhko Školitel: prof. PhDr. Oldřich Uličný, DrSc. Rozšírená textová Koreference a asociační anafora (Koncepce anotace českých dat v Pražském závislostním korpusu) Univerzita Karlova v Praze Filozofická fakulta Ústav českého jazyka a teorie komunikace, 2010 (M.Chytil, 1984)

[9] http://ufal.mff.cuni.cz/pdt2.0/

[10] Lucie Kučová, Veronika Kolářová-Řezníčková, Zdeněk Žabokrtský, Petr Pajas, Oliver Čulo: "Anotování koreference v Pražském závislostním korpusu". V: *ÚFAL Technical Report*, 19, MFF UK, Prague, Czech Republic, 2003.

[11] Lucie Kučová, Eva Hajičová: "Coreferential Relations in the Prague Dependency Treebank". V: (ed.): *Proceedings of the 5th International Conference on Discourse Anaphora and Anaphor Resolution 2004*, San Miguel, Azores, Sept. 23-24, 2004, 2005, pp. 94-102.

[12] Jarmila Panevová, Eva Hajičová, Petr Sgall: "Coreference in Annotating a Large Corpus". V: M. Gavrilidou, G. Carayannis, S. Markantonatou, S. Piperidis, G. Stainhaouer (eds.): *Proceedings of the 2nd International Conference on Language Resources*, (I), European Language Resources Association, Athens, Greece, 2000, pp. 497-500.

[13] Lucie Kučová and Zdeněk Žabokrtský. 2005. Anaphorain Czech: Large Data and Experiments with Automatic Anaphora. In LNCS/Lecture Notes in Artificial Intelligence/Proceedings of Text, Speech and Dialogue.Springer Verlag Heidelberg.

[14]Shalom Lappin and Herbert J. Leass. 1994. □ an algorithm for pronominal anaphora resolution□ . Computational Linguistics,20(4):535□ 561.

[15]Nguy Giang Linh. 2006. Proposal of a Set of Rules for Anaphora Resolution in Czech. Master's thesis, Faculty of Mathematics and Physics, Charles University.

[16] http://ufal.mff.cuni.cz/~pajas/tred

[17] Barbora Hladká, Ji°í Mírovský, Jan Kohout, The Prague Bulletin of Mathematical Linguistics, JULY 2011 1-23, An attractive game with the document: (im)possible?

[18] http://nlp.fi.muni.cz/projekty/bonito/bonito.html.cz

[19]http://ufal.mff.cuni.cz/rest/CAC/doccac10/cacguide/cz/html/kapitola3.html#nastroje

Príloha A- Pseudokód

```
Zoznam Premenne AplikujAutomatNaText(Zoznam Slov
Text,String Pravidlo,Premenne P)
{
  Zoznam Automat:
AutomatyPovodne=VytvorAutomatyPodlaPravidla(Pravidlo);
  If(P.IsNotEmpty())
  {
    (foreach Automat AA in AutomatyPovodne)
    {
      AA.VlozPremenne(P);
    }
  }
  FIFO Fronta Automat:Automaty
  Zoznam Premenne: Vysledok
  For (i=0;i<Pocet Slov,i++)
  {
     Automaty.Push(AutomatyPovodne)
     PocetAutomatov=Automaty.Dlzka
     For(j=0;j<PocetAutomatov;j++)
     {
       Automat A=Automaty.front();
       If(JeVKoncovomStave(A))
       {
        Vysledok.Add(A.Premenne);
       }
     }
    Automaty.PushAll(VlozSlovo(Text(i)),Automaty[i]))
    Automaty.pop();
  }
 return Vysledok
}
```