VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

# SYNTAKTICKÁ ANALÝZA ZALOŽENÁ NA STAVOVÝCH GRAMATIKÁCH

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MIROSLAV NOVOTNÝ
AUTHOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

# SYNTAKTICKÁ ANALÝZA ZALOŽENÁ NA STAVOVÝCH GRAMATIKÁCH

PARSING BASED ON STATE GRAMMARS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE Miroslav Novotný
AUTHOR

VEDOUCÍ PRÁCE Meduna Alexander, prof. RNDr., CSc.
SUPERVISOR

BRNO 2015

# Abstrakt

Tato práce se zabývá syntaktickou analýzou založenou na stavových gramatikách. Cílem je vytvořit program schopný načíst gramatiku ze vstupního souboru. Na základě této gramatiky vytvořit LL tabulku a následně i provést syntaktickou analýzu zadaného vstupu. Na těchto základech pak studovat vlastnosti metod syntaktické analýzy, založené na těchto gramatikách. Testování probíhá i na gramatických strukturách, které nejsou bezkontextové.



# Abstract

This thesis's main focus is parsing, based on state grammars. The goal is to create a program, that will be able to load the grammar from input file. Based on a loaded grammar, the program will create an LL table and parse an input file using this table. The next goal is to study properties of parsing, based on state grammars, while using a created program as a stand point. Part of the testing will also be grammar structures which are not context-free.



# Klíčová slova

Regulované gramatiky, stavové gramatiky, LL tabulka, syntaktická analýza

# Keywords

Regulated grammars, state grammars, LL table, parsing


# Citace

Novotný Miroslav: Syntaktická analýza založená na stavových gramatikách, bakalářská práce, Brno FIT VUT v Brně 2015

# Syntaktická analýza založená na stavových gramatikách

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, pod vedením pana profesora Alexandera Meduny.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Miroslav Novotný
11. 5. 2015

## Poděkování

Tímto děkuji panu profesoru Medunovi, za vedení a všechnu pomoc v rámci mé bakalářské práce.

# Obsah

# 1 Úvod

Tato práce spadá v rámci informačních technologií do oblasti formálních jazyků. Tento obor se zabývá teorií jazyků a jejich matematickou formalizací. Jazyk, jako takový, je velmi důležitým nástrojem předávání informací. Analýzou přirozeného jazyka se však, pro jeho přílišnou složitost, nebudeme v této práci zabývat. Do oblasti formálních jazyků spadají též programovací jazyky, jejichž překladače a interprety stojí na poznatcích z teorie formálních jazyků.

Záměrem této práce je prostudovat vlastnosti a chování stavových gramatik, jejich použití v různých situacích a také ve spolupráci s dalšími aparáty. Stavové gramatiky, jsou druh regulovaných gramatik. Regulované gramatiky jsou založeny na klasický gramatikách, tedy jsou jejich rozšířením, které dokáže kontrolovat výběr pravidla použitého v rámci generování jazyka.

Cílem je také zjistit, zda a jak se hodí spojit stavové regulované gramatiky s analýzou založenou na LL tabulkách. V rámci implementace jsem začal s postupem LL(1) a postupně přidával další aparáty, které měly za úkol zvýšit sílu gramatik a zároveň otestovat vhodnost využití tohoto spojení.

V rámci této aplikace je předpokládána kompletní determinističnost pravidel. Tento fakt se může ukázat jako omezující faktor co se týče síly gramatik, ale je nutný pro správnou funkci analýzy založené na nejlevější derivaci.

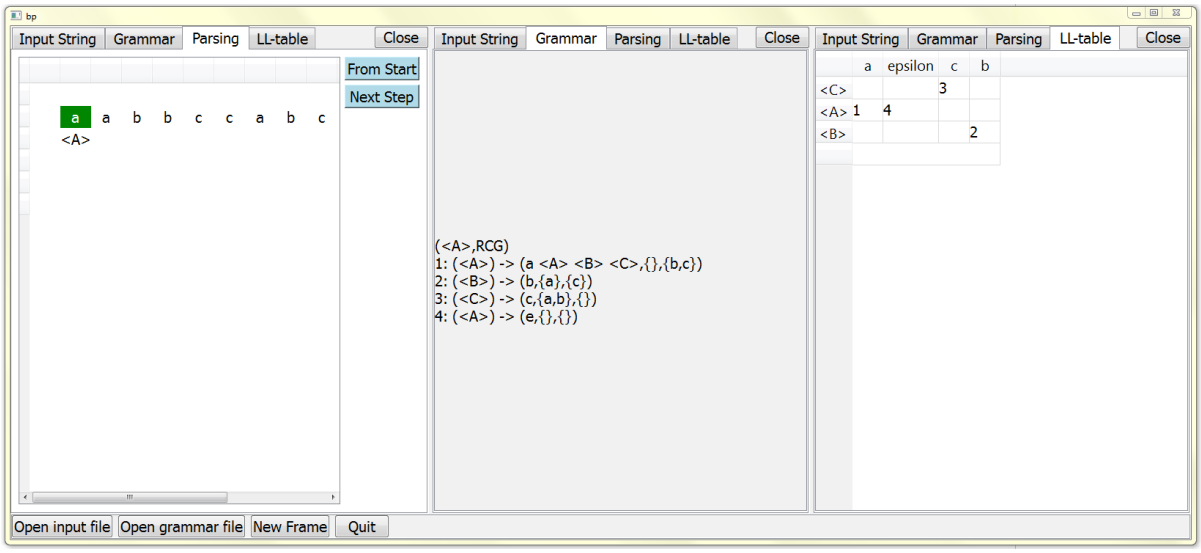
Jako rozšíření uvažujeme například úpravu LL analýzy ze striktně nejlevějšího na nejlevější možný vstupní znak, pro který máme pravidlo. Předpokladem je, že tato úprava zvýší množství přijatých jazyků beze změny na straně gramatiky. Riziko tohoto přístupu ovšem je ztráta kontroly, neboť nenabízí dostatek pro pravidla restriktivních prostředků. Z toho důvodu zavádíme další rozšíření, která má za úkol zvýšit kontrolu přidání omezujících faktorů do jednotlivých pravidel.
Tímto aparátem jsou gramatiky s nahodilým kontextem (anglicky *Random Context Grammars)*. Pomocí těchto úprav jsme schopni silně zvýšit naši kontrolu nad generovaným jazykem, zavedením restriktivních podmínek do pravidel v naší gramatice.

Jelikož součástí práce je i grafická reprezentace výše uvedených postupů, jako další prvek zavádíme uživatelské rozhraní. Toto rozhraní obsahuje nástroje, které umožňují přehlednou prezentaci syntaktické analýzy a struktur k tomu použitých. Cílem také je, aby bylo rozhraní přehledné a snadno ovladatelné. Jedna z vlastností programu je též možnost přizpůsobit se změnám velikosti okna a změně množství zobrazovaných informací.

Tato práce se skládá z několika kapitol. Jako první uvádím Teorie a definice. V této kapitole, jak již název napovídá, probereme základní použité elementy z teorie formálních jazyků, jako jsou například jazyk a gramatika. Následující část je věnována postupu analýzy při využití striktně nejlevější derivace. Tento postup je často používaný a nejen proto velmi důležitý. Třetí kapitolou, nepočítám-li úvod, je teorie o stavových gramatikách, poněvadž je prostudování těchto gramatik jedním z hlavních cílů této práce, je nutné s ní čtenáře seznámit. Další následuje seznámení se s aparátem nejlevější možné derivace, který rozšiřuje postup syntaktické analýzy a tím i možnosti gramatik co se týče přijatých řetězců. Část, která následuje poté, se věnuje tématu gramatik s nahodilým kontextem.

Tento přídavek má za účel rozšířit možnosti programu, jak co se týče typů gramatik, které dokáže převést na LL tabulky, tak co se týče rozsahu jazyků, které dokáže přijmou syntaktickým analyzátorem, při použití těchto tabulek. Kapitola grafická reprezentace pojednává o způsobu, jakým jsou informace předávány uživateli programu a také o možnostech jeho ovládání. V kapitole implementace jsou základní informace o struktuře programu a také v jakém jazyce a s jakými knihovnami byl vytvořen. V předposlední části s názvem výsledky a testování je předvedeno základní použití programu a jsou tam uvedeny závěry, ke kterým práce míří. V závěru textového dokumentu jsou napsány přínosy této práce a také jsou zde uvedeny možnosti případného rozšíření.

Ukázka z výsledného programu:



# 2 Teorie a definice

V této kapitole se seznámíme se základy formálních jazyků a s některými dalšími aparáty týkající se této práce. Všechny definice jsou převzaté z [4].

## 2.1 Základní definice

Abeceda je konečná neprázdná množina elementů zvaných symboly.
Slovo nebo řetězec nad abecedou $\Sigma$ je konečná posloupnost symbolů z $\Sigma$.
Prázdný řetězec je označován jako $\varepsilon$.
Jazyk L nad $\Sigma$ je množina řetězců nad $\Sigma$. Množinu všech řetězců nad $\Sigma$ nazýváme univerzální jazyk.
Terminál je jeden znak z řetězce z daného jazyka. Neterminál je typ symbolu, který může být v rámci analýzy nebo generování přepsán dle vhodného pravidla.

## 2.2 Gramatiky

Gramatika je čtveřice $G = (N,T,P,S)$, kde

1) N je abeceda neterminálů
2) T je abeceda terminálů, přičemž platí $N \cap T = \emptyset$;
3) $P \subseteq (N \cup T)^*N(N \cup T^*) \times (N \cup T)^*$ je konečná množina přechodů;
4) $S \in N$ a značí počáteční neterminál

Páry $(u,v) \in P$, nazýváme přepisovacími pravidly

## 2.3 Regulované gramatiky

Regulovaná gramatika je bezkontextová gramatika rozšířena o další přídavný aparát, který určuje jaká pravidla budou použita v rámci generování jazyka.

## 2.4 LL-analýza

V rámci LL analýzy dochází k postupu zleva doprava a konstrukce nejlevější derivace řetězce. Tato analýza je založena na LL tabulce, její struktura je uspořádána tak, aby pro každý jeden terminál a neterminál existovalo právě jedno pravidlo z klasické gramatiky. V případě stavových gramatik, existuje právě jedno pravidlo pro jeden terminál, jeden neterminál a jeden stav současně.

Analýza probíhá tak, že se postupně aplikuje pravidlo na vrchní položku v zásobníků, v níž se na začátku nachází počáteční neterminál, a nejlevější vstupní symbol (*token*). Při aplikaci pravidla dochází k rozgenerování jeho pravé strany v opačném pořadí na vrchol zásobníků. Tímto způsobem se pokračuje dokud není zásobník prázdný. Pokud dojde k vyprázdnění zásobníků, ale vstupní řetězec není ještě zpracován celý, pak řetězec není syntaktickým analyzátorem přijat. Stejně tak pokud již dojde vstup, ale zásobník ještě není prázdný. Řetězec je úspěšně přijat pouze tehdy, dojde-li současně k vyprázdnění zásobníku a dočtení celého vstupního řetězce.

# 3 Striktně nejlevější derivace

Také bývá nazývána nejlevější derivace typu 1. V rámci každého derivačního kroku dochází k přepsání nejlevějšího výskytu neterminálu.
Tento přístup k syntaktické analýze, kdy za pomoci LL tabulky probíhají derivace v přesně daném pořadí, umožňuje jednoduchou implementaci a transparentní průběh analýzy. Tento postup je často používán při tvorbě programovacích jazyků.
Tvar LL tabulky přímo závisí na daný pravidlech, k jejímu vytvoření tedy stačí pouze samotná pravidla a algoritmus pro převedení založený na množinových operacích. Klasické gramatiky za použití LL(1) analýzy jsou schopny přijímat bezkontextové jazyky.
V rámci práce budeme uvažovat tento tvar pravidel:
[číslo pravidla]: [levý neterminál] → [sekvence terminálů a neterminálů]
    například 1: <A> → a b c <A>

Jako příklad použijeme jazyk $a^n b^n$.
Klasická gramatika se může skládat například z pravidel:
    1: <S> → a <S> b
    2: <S> → ε

Kde <S> je počáteční neterminál.

Pro tuto gramatiku existuje LL tabulka:

| | a | ε |
|---|---|---|
| <S> | 1 | 2 |

Pak například řetězec aaaabbbb generujeme použitím posloupnosti pravidel 1,1,1,1,2 .

Pro řetězec aabb by probíhala analýza takto:

| | | |
|---|---|---|
| Krok 1. | **a** a b b | Začátek syntaktické analýzy. |
| | <S> | Původní stav. |
| Krok 2. | **a** a b b | Rozgenerování neterminálu <S>. |
| | a <S> b | Použití pravidla 1. |
| Krok 3. | **a** b b | Přijetí terminálu 'a'. |
| | <S> b | |
| Krok 4. | **a** b b | Rozgenerování neterminálu <S>. |
| | a <S> b b | Použití pravidla 1. |
| Krok 5. | **b** b | Přijetí terminálu 'a'. |
| | <S> b b | Rozgenerování neterminálu <S>. |
| Krok 6. | **b** b | Použití pravidla 2. |
| | b b | |
| Krok 7. | **b** | Dvě následné přijetí terminálu 'b'. |
| | b | |

Syntaktická analýza proběhla úspěšně řetězec patří do jazyka generovaného výše zmíněnou gramatikou.

# 4 Stavové gramatiky

V této kapitole se seznámíme se stavovými gramatikami, jejich silou a výsledkem jejich kombinace s LL analýzou.

Regulované gramatiky jsou obecně snahou o zvýšení množství jazyků generovaných danými gramatikami, pomocí přidání dalšího řídícího aparátu, který by umožňoval ovlivňovat jaké pravidlo bude použito a kdy. V případě stavových gramatik, je do gramatik přidán systém, připomínající konečný automat. V rámci každého provedení nějakého pravidla dojde kromě přepisu neterminálu i ke změně stavu a tedy k ovlivnění výběru dalšího pravidla.

Stavová gramatika je pětice G = (V, W, T, P, S), kde

1) V je celková abeceda
2) W je konečná množina všech stavů
3) T je abeceda terminálů, přičemž $T \subseteq V$
4) S je počáteční symbol, $S \in V - T$
5) $P \subseteq (W \times (V\text{-}T)) \times (W \times V^{+})$

Ač je generativní síla stavových gramatik na úrovni kontextových jazyků, při aplikování striktně nejlevější analýzy, přijímají stavové gramatiky pouze bezkontextové jazyky.

Za snížení síly v tomto případě může jak striktní nutnost determinismu pravidel, tak nemožnost provést přepsání jiného neterminálu, než toho na vrcholu zásobníku.

V rámci práce a programu s ní spjatém pracuji s pravidly v tomto tvaru:
[číslo pravidla]: ([výchozí stav],[levý neterminál]) → ([stav po aplikaci pravidla],[sekvence terminálů a neterminálů])

například 1: (start,<A>) → (end,a b c <A>)

Jako příklad použijeme jazyk $a^n b a^n$.
Gramatika za použití stavových pravidel:

1: (1,<A>) → (1,a <A> <A>)
2: (1,<A>) → (2,b)
3: (2,<A>) → (2,a)

Kde <A> je počáteční neterminál.

| Stav 1 | a | b | Stav 2 | a |
|---|---|---|---|---|
| <A> | 1 | 2 | <A> | 3 |

Pro řetězec a a a a b a a a a budou použita pravidla 1,1,1,1,2,3,3,3,3 .

Definice převzatá z [3].

# 5 Nejlevější možná derivace

V této kapitole rozšíříme možnosti analýzy o nový aparát. Na místo odebrání nejlevějšího symbolu na vstupu k nalezení pravidla použijeme nejlevější možný symbol, pro který existuje pravidlo. Metoda načtení prvního vhodného symbolu ze vstupního řetězce byla použita po konzultaci s vedoucím práce. Jedná se o úpravu metody, ve které se na místo striktně nejlevějšího rozgeneruje nejlevější možný neterminál na zásobníku automatu.

## 5.1 Použití

S tímto aparátem dochází při výběru pravidla k procházení vstupního řetězce zleva dokud není nalezen symbol, pro který existuje pravidlo. V případě, že algoritmus dojde až na konec řetězce, je možné ještě použít pravidlo s řetězcem nulové délky neboli ε.
Jelikož se rozdíl nachází v algoritmu syntaktické analýzy, nedochází k žádné změně tvaru načítaných pravidel nebo jejich převedení na LL tabulku.

## 5.2 Příklad s $a^nb^nc^n$

Jazyk $a^nb^nc^n$ bývá často používán jako příklad jazyka, který není bezkontextový, ale kontextový. Klasické gramatiky nebo i stavové gramatiky za použití LL(1) analýzy na něj tedy nestačí. V případě rozšíření LL(1) o hledání nejlevější možné derivace, je možné napsat gramatiku, která tento jazyk generuje.
Tato gramatika se sestává například s pravidel:
1: <A> → a <B>
2: <B> → b <C>
3: <C> → c <A>
4: <A> → epsilon
Kde <A> je počáteční neterminál.
Těmto pravidlům odpovídá LL(1) tabulka:

| | a | b | c | ε |
|---|---|---|---|---|
| <A> | 1 | | | 4 |
| <B> | | 2 | | |
| <C> | | | 3 | |

Pak budou pro řetězec a a a a a b b b b b c c c c c použita pravidla:
1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 4

## 5.3 Problémy tohoto přístupu

S pomocí této gramatiky můžeme generovat jazyk $a^nb^nc^n$, stejně jako můžeme přijmout řetězec tohoto jazyka při syntaktické analýze. Tento přístup nám ovšem neumožňuje dostatek restriktivní moci k tomu, abychom přijímali gramatikou pouze tento jazyk.
Gramatika uvedená v předchozím příkladu sice dovede přijmout $a^nb^nc^n$, přijímá ovšem i další řetězce. Abychom byli konkrétní, tato gramatika akceptuje všechny řetězce, ve kterých se vyskytuje stejný počet symbolů 'a', 'b' a 'c'. Jazyk $a^nb^nc^n$ je tedy podmnožinou jazyka generovaného touto gramatikou.

# 6 Další řídicí aparát

V této kapitole uvedeme možnosti použití gramatik s nahodilým kontextem, neboli *Random Context Grammars* pro zvýšení restriktivní síly regulace pravidel.
Informace a definice čerpány z [6].

## 6.1 Definice

V rámci tohoto aparátu dochází k rozšíření, ve kterém je ke každému pravidlu přidána množina přikázaných a množina zakazujících symbolů.

Gramatika s nahodilým kontextem je pětice. $G = (N, T, P_L, P_R, S)$, kde

1) N je abeceda neterminálů
2) T je abeceda terminálů
3) $P_L$ a $P_R$ jsou dvě konečné množiny pravidel ve tvaru
   $[A \rightarrow x, U, W]$
   kde $A \in N$, $x \in (N \cup T)^*$, a $U,W \subseteq N$;
4) S je počáteční neterminál

## 6.2 Použití

Tento přístup umožňuje přesnější omezení množiny generovaných jazyků. Řeší tedy problém, na který jsme narazili v minulé kapitole. Zatímco s analýzou s nejlevějším možným akceptovatelným symbolem dokážeme přijmout $a^nb^nc^n$, ale nedokážeme přijmout *pouze* $a^nb^nc^n$.
Pravidlo obsahuje dvě množiny zakazujících terminálů. První množina určuje, které terminály se musí vyskytovat před místem použití pravidla a nesmí se vyskytovat za ním. Druhá množina naopak přikazuje jaké terminály se musí vyskytovat za místem použitím pravidla, a které se nesmí vyskytovat před.

Použitý tvar pravidel je:
[číslo pravidla]: ([levý neterminál]) → ([sekvence terminálů a neterminálů],[{množina terminálů}], [{množina terminálů}])
Příklad:
2:(<A>) → (b, {a},{c})

nebo v případě stavové gramatiky:
[číslo pravidla]: ([výchozí stav],[levý neterminál]) → ([stav po aplikaci pravidla],[sekvence terminálů a neterminálů],[{množina terminálů}],[{množina terminálů}])
Příklad:
2:(p0,<A>) → (p1,b, {a},{c})

## 6.3 Příklad s $a^n b^n c^n$

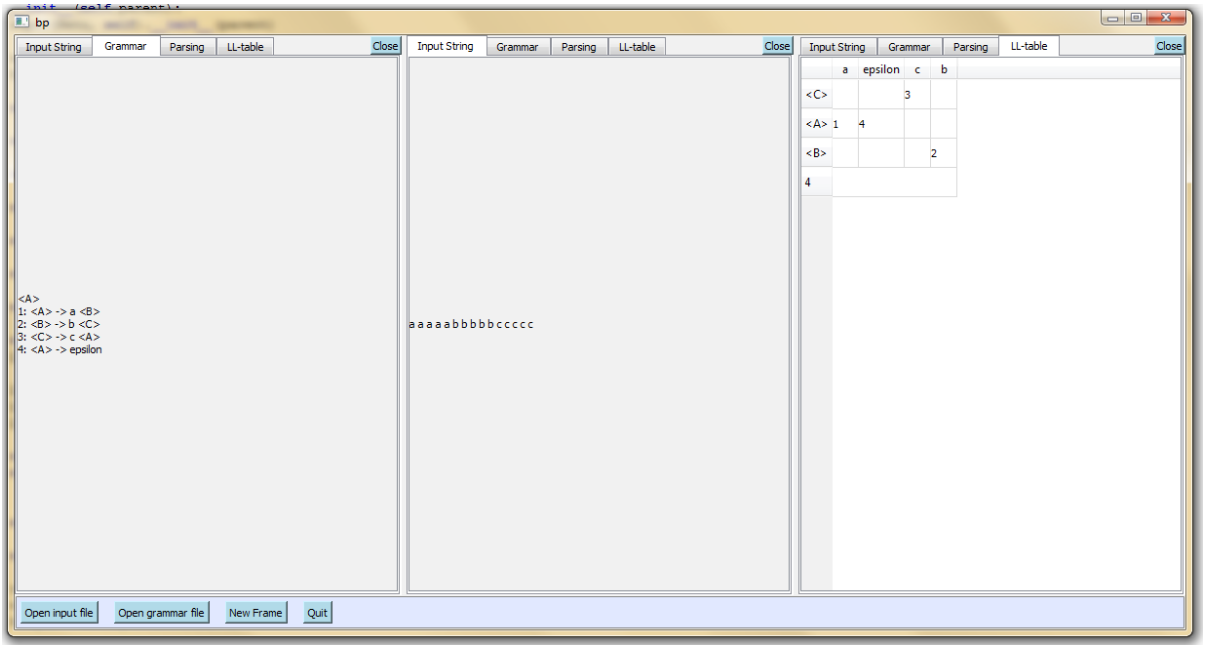
Test bude probíhat s řetězcem a a b b c c a b c, který neodpovídá jazyku $a^n b^n c^n$. Tento řetězec by však byl bez použití gramatik s nahodilým kontextem přijat.
Testovací gramatika:
(<A>,RCG)
1:(<A>) → (a <A> <B> <C>, {},{b,c})
2:(<A>) → (b, {a},{c})
3:(<A>) → (c, {a,b},{})
4:(<A>) → (ε, {},{})

Syntaktická analýza se zastaví s neúspěchem po použití pravidel v tom to pořadí: 1, 1, 4, 2, 3, 2, 3. Zásobník je již prázdný a na vstupu stálé zůstávají znaky 'a', 'b' a 'c'.

# 7 Grafická reprezentace

Součástí bakalářské práce je také grafická reprezentace . Cílem bylo vytvořit uživatelské rozhraní umožňující výběr vstupních souborů, prohlédnutí syntaktických struktur a přehledné krokování průběhu syntaktické analýzy.

## 7.1 Celkový pohled

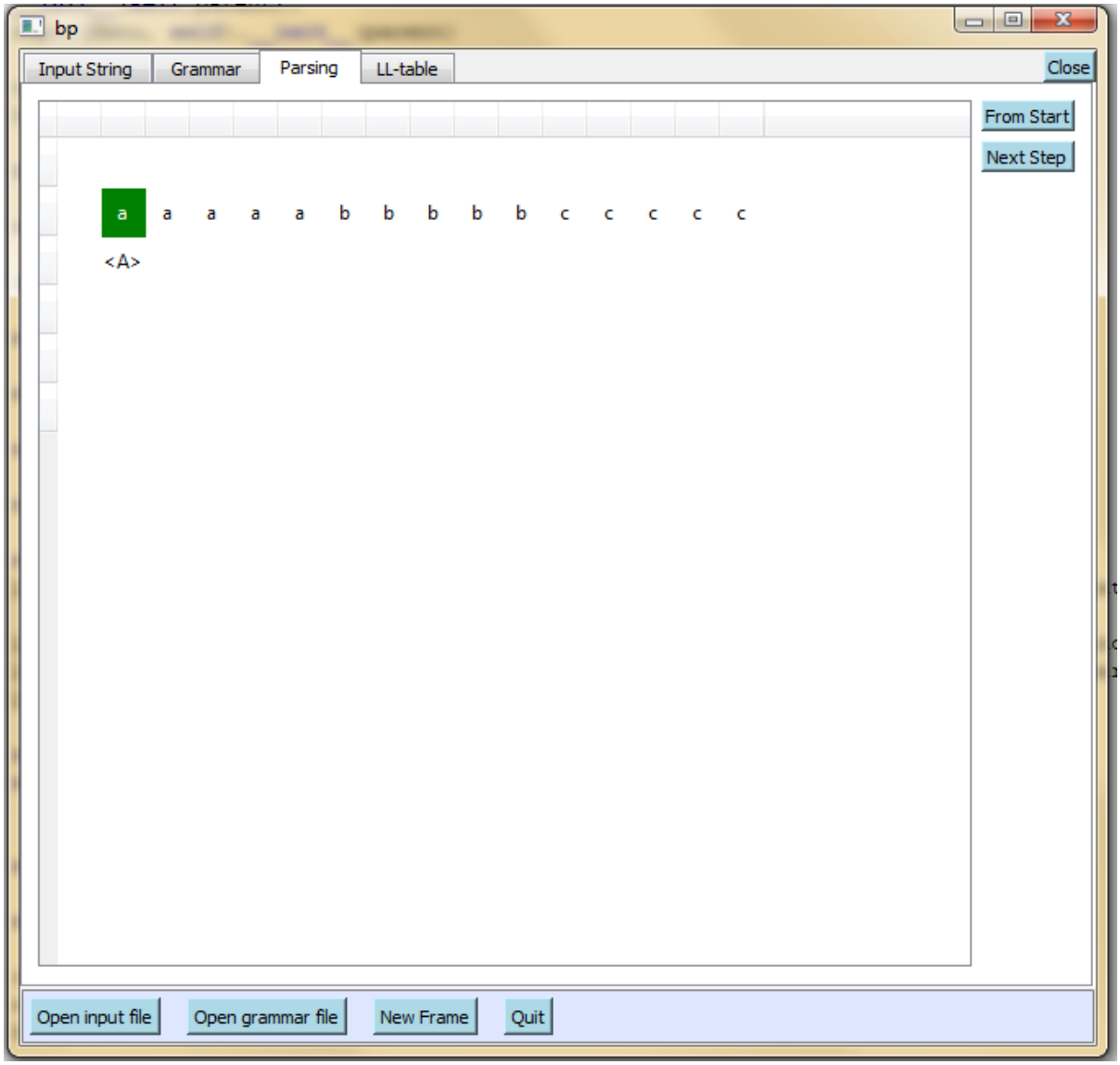



Zde je příklad, jak vypadá program za běhu, v následujících odstavcích popíši jednotlivá okna, nejprve však spodní panel. Na řádku dole pod hlavními okny programu se nacházejí tři tlačítka. První dvě tlačítka, `Open input file` a `Open grammar file`, otevřou nové okno, které umožní načíst nový vstupní soubor, řetězec jazyka nebo gramatiku. Následující tlačítko `New Frame`, vloží do hlavního okna nový panel, který v sobě bude obsahovat nejnovější načtenou gramatiku a vstupní řetězec a bude se na něm dát zvolit jeho obsah. Poslední tlačítko `Quit`, ukončí program.

V rámci každého panelu, neboli rámu, jsou čtyři možnosti, měnící typ obsahu, který je v něm zobrazený. Jsou to části `Input String`, `Grammar`, `Parsing` a `LL-table`.

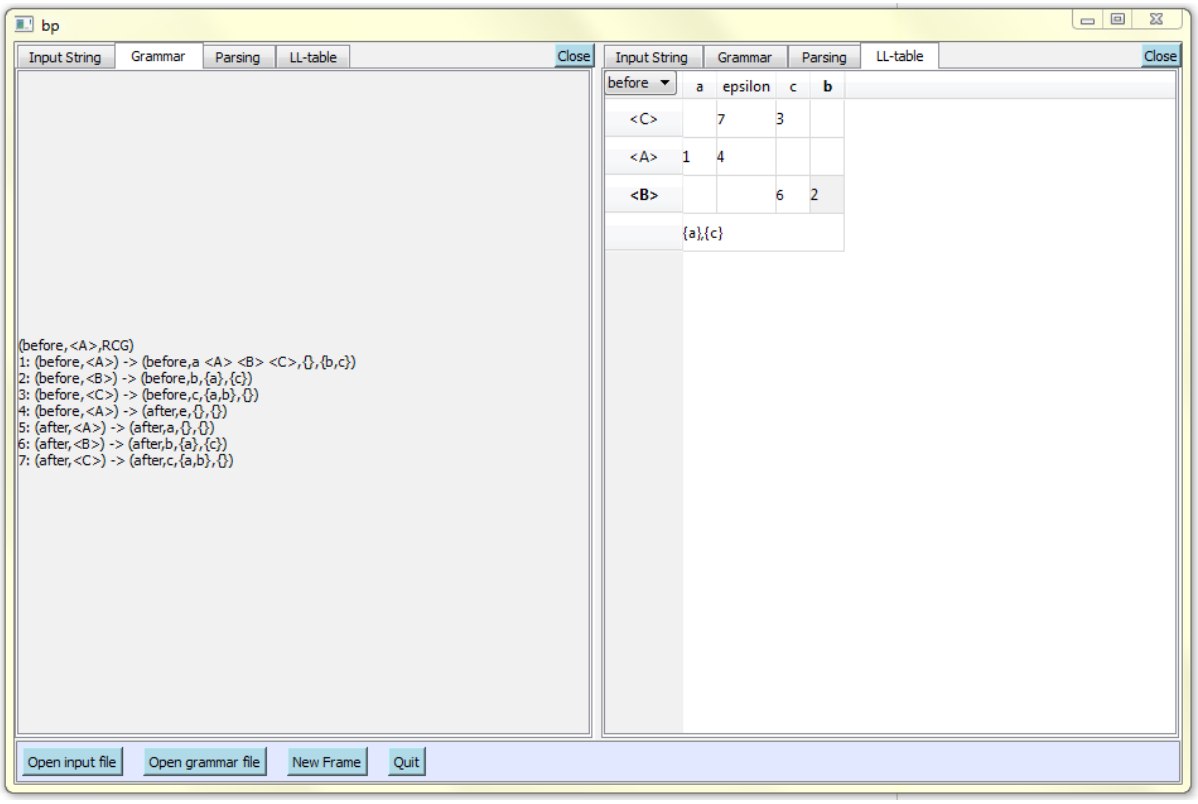
## 7.2 Panel Parsing

V tomto panelu probíhá krokování běhu syntaktické analýzy, na prvním řádku je vstupní řetězec, na druhém řádku je stav zásobníku v průběhu analýzy. V tomto případě ještě neproběhl ani jeden krok analýzy, na vrcholu zásobníku je počáteční neterminál a vstupní řetězec je zobrazen celý.




Jak je vidět, na pravé straně panelu se nacházení dvě tlačítka `From Start` a `Next Step`. První zmíněné vrátí stav krokování do stavu, ve kterém se vyskytoval na začátku. Druhé tlačítko provede jeden krok syntaktické analýzy.

## 7.3 Panely Gramatika a LL tabulka




Na tomto obrázku vidíme dva panely, v panelu vlevo je zobrazená načtená gramatika. V pravé části obrázku je LL tabulka, která byla vygenerována na základě gramatiky. Jednotlivá políčka tabulky se dají označit pro další informace. V levém horním rohu tabulky se nachází přepínač stavu, pro který platí daná pravidla. Zobrazená tabulka se změní při změně tohoto stavu.

## 7.4 Vstupní řetězec

Poslední typ obsahu panelu je vstupní řetězec, pokud je vybrán, na obrazovce se objeví v tom tvaru, v jakém byl načten ze souboru.

# 8 Implementace

Celý program bakalářské práce je implementován v jazyce Python (2.7) a pro zobrazení uživatelského rozhraní je použita knihovna pyQt (4).

## 8.1 Rozdělení programu

Program je modulárně rozdělen na několik částí. Několik části se stará o funkci uživatelského rozhraní, jako například položky menu. Uživatelské rozhraní je pro větší přehlednost ještě rozděleno do samostatných souborů podle funkčnosti. Další část je pro načtení vstupní gramatiky a její převedení do tabulky. Následně je tu modul starající se o syntaktickou analýzu. Poslední dvě části jsou konfigurační soubor, sloužící pro snadnější řízení některých částí programu, jako je například počáteční velikost okna, a modul s datovými strukturami, pro uložení například gramatiky nebo načteného vstupního souboru.

## 8.2 Načtení gramatiky

Zpracování gramatiky ze souboru je iniciováno z řídicí části programu. Tato část je úzce spojena s uživatelským rozhraním. Samotná proces načtení gramatiky začíná otevřením souboru zadaného cestou. Po úspěšném otevření souboru je analyzován první řádek a poté je rozhodnuto jak se bude ke gramatice přistupovat. Obsahem úvodního řádku souboru jsou povinné informace, nutné nejen k rozpoznání, ale také k následnému běhu konverze na LL tabulku.

Příklad: (<A>)
V tomto případě je zadán pouze počáteční neterminál, jedná se tedy o klasickou gramatiku.

Příklad: (base,<var>)
Toto zadání odpovídá stavové gramatice, kromě počátečního neterminálu je určen i počáteční stav.

V obou výše zmíněných případech může být dále přidáno označení `RCG` (zkratka z `Random Context Grammars`), které značí, že bude použito aparátu Gramatik s nahodilým kontextem.

Příklady: (before,<A>,RCG) nebo (<A>,RCG)

Postup při převodu gramatiky na LL tabulku je následující. Ze souboru jsou načtena jednotlivá pravidla, ty jsou rozpoznána pomoci regulárních výrazů a následně, pomocí algoritmu založeném na množinových operacích, převedena. V této části jsou též rozpoznány a zahrnuty do struktury tabulky stavové části pravidel a případně `RCG` pravidla.

## 8.3 Syntaktická analýza

V programu je implementována metoda pro jeden krok syntaktické analýzy. Tato metoda je opakovaně volána ze strany uživatelského rozhraní. Funkce se snaží primárně použít striktně nejlevějšího vstupního symbolu a v případě nenalezení pravidla přechází do režimu hledání nejlevějšího možného symbolu.

## 8.4 Grafické rozhraní

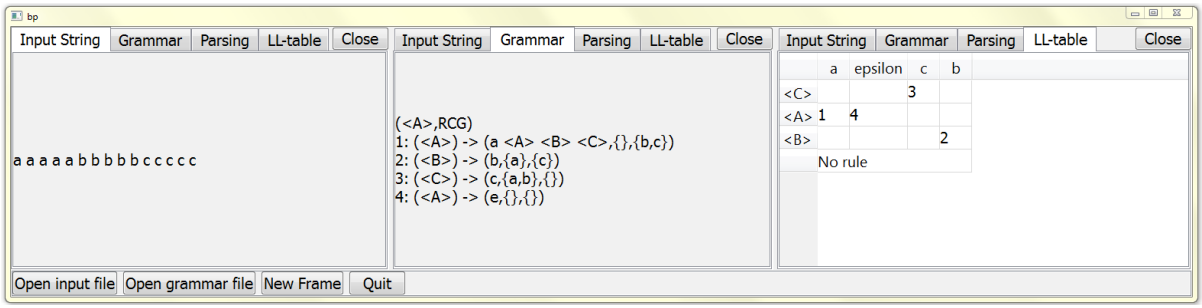
Každá grafická třída v rámci programu dědí z nějakého základního prvku z knihovny pyQt. Hierarchie tříd tvoří strukturu rozhraní.

# 9 Výsledky a testování

Nejpodstatnější informace, která plyne z výsledků je, že samotné stavové gramatiky nejsou při využití LL analýzy dostatečně mocné, aby dokázaly přijímat kontextové jazyky. I proto byla práce rozšířena o přídavné aparáty, které umožnily zvýšit sílu gramatik a dosáhnout původní předpokládané síly.

Program, který je výsledkem práce, je schopný načíst gramatiku a s její pomocí syntakticky analyzovat kontextový jazyk, jenž dané gramatice odpovídá.

Na následujícím obrázku máme příklad spuštění programu pro gramatiku, která obsahuje gramatiku s *RCG* rozšířením. Tato gramatika přijímá pouze jazyk $a^n b^n c^n$.




Pozn. Velikost písma byla v rámci testování zvětšena, aby byl následný výstup čitelnější.

Postup výpisu programu je dále následující:

Krok 1. Původní vstupní řetězec.
**a** a a a a b b b b b c c c c c Počáteční neterminál.
<A>
Krok 2. Použití pravidla 1.
**a** a a a a b b b b b c c c c c
**a** <A> <B> <C>
Krok 3.
**a** a a a b b b b b c c c c c
<A> <B> <C>
Krok 4. Použití pravidla 1.
**a** a a a b b b b b c c c c c
**a** <A> <B> <C> <B> <C>

… Analýza dále pokračuje obdobným způsobem.

Krok 11.
**b** b b b b c c c c c
<A> <B> <C> <B> <C> <B> <C> <B> <C> <B> <C>
Krok 12.
**b** b b b b c c c c c
**epsilon** <B> <C> <B> <C> <B> <C> <B> <C> <B> <C>
Krok 13.
**b** b b b b c c c c c
<B> <C> <B> <C> <B> <C> <B> <C> <B> <C>
Krok 14.
**b** b b b b c c c c c
**b** <C> <B> <C> <B> <C> <B> <C> <B> <C>
Krok 15.
**b** b b b c c c c c
<C> <B> <C> <B> <C> <B> <C> <B> <C>
Krok 16.
b b b b **c** c c c c
**c** <B> <C> <B> <C> <B> <C> <B> <C>
Krok 17.
**b** b b b c c c c
<B> <C> <B> <C> <B> <C> <B> <C>
…
Krok 33.
Syntaktická analýza úspěšná, použitá pravidla: 1, 1, 1, 1, 1, 4, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3

# 10 Závěr

V rámci práce jsem zjistil několik zajímavých informací o použití stavových gramatik. Kombinace s LL analýzou omezuje jejich sílu. Mohlo by se tedy zdát, že tím nemá toto spojení žádný praktický smysl. Z jiného hlediska však tyto gramatiky přinášejí řadu výhod. První, a dost možná nejdůležitější, výhodou je zvýšení přehlednosti gramatik a syntaktické analýzy. Přítomnost stavů funkčně odděluje části gramatiky a LL tabulky zpřehledňuje jejich rozdělením na části. Tento význam se dá přirovnat k modulárnímu přístupu v programování. Další výhodou je usnadnění zápisu deterministických pravidel, zvláště za použití gramatik s nahodilým kontextem se může stát, že se pravidla budou v jistých případech chovat nedeterministicky. Tyto případy je možné částečně obcházet za použití stavových pravidel.
Na druhou stranu, stavové gramatiky nijak významně nepřispěly v úkolu vytvoření syntaktického analyzátoru použitelného pro kontextové jazyky. Těchto vlastností bylo nakonec dosaženo pomocí jiných rozšíření než stavových gramatik.
Metoda syntaktické analýzy, založena na metodě LL analýzy typu 2, se ukázala jako mocný nástroj z hlediska rozšíření množiny řetězců přijímanými danou gramatikou. Tato metoda ovšem postrádala dostatečné kontrolní mechanismy k dosažení jednoho z cílů a to vytvoření gramatiky, která by přijímala pouze jazyk $a^n b^n c^n$.
Tento problém byl nakonec vyřešen přidáním restriktivních pravidel z gramatik s nahodilým kontextem.

Program, který je součástí práce obsahuje všechny původně plánované prvky. Tedy hlavně možnost přehledného krokování postupu syntaktické analýzy, ale co se týče zobrazení dat programem využívaných. V rámci implementace grafického rozhraní jsem se, co se týče panelů, inspiroval moderními internetovými prohlížeči.

Jako další možné pokračování práce by bylo možné rozšířit možnosti programu z hlediska uživatelského přístupu. Například zvýšit možnosti krokování pomocí bodů přerušení nebo umožnit upravovat struktury programu za jeho běhu.
Zajímavým rozšiřujícím tématem by také mohlo být větší řízení programu gramatikou. Lexikální analyzátor nebyl součástí této práce, mohl by proto být také řízen gramatikou.
Co se týče oblasti teorie jazyků, práce by mohla pokračovat detailním prostudováním kombinace RCG se stavovými gramatikami, nebo o rozšíření popisu kombinace stavových gramatik s LL přístupem o matematické důkazy.

# Literatura

[1] Rozenberg, G. And Salomaa, A. (eds.): Handbook of Formal Languages, Volume 1 through 3, Springer, 1997, ISBN 3-540-60649-1

[2] Aho, A. V.,Sethi, R., Ullman, J. D. : Compilers : principles, techniques, and tools, Addison-Wesley, 2nd ed., 2007, ISBN: 0321486811

[3] MEDUNA, Alexander a Petr ZEMEK. 2010. *Regulated Grammars and Their Transformation*. První. Brno: Vysoké učení technické v Brně Fakulta informačních technologií. ISBN 978-80-214-4203-0.

[4] MEDUNA, Alexander a Petr ZEMEK. 2014. *Regulated Grammars and Automata*. New York: Springer. ISBN 978-1-4939-0369-6.

[5] PyQt4 Reference Guide. 2014. *PyQt4 Reference Guide* [online]. [cit. 2015-05-13]. Dostupné z: http://pyqt.sourceforge.net/Docs/PyQt4/index.html

[6] MEDUNA, Alexander, Lukáš VRÁBEL a Petr ZEMEK. 2012. *One-Sided Random Context Grammars* [online]. [cit. 2015-05-14]. Dostupné z: http://www.fit.vutbr.cz/~izemek/grants.php.cs?file=%2Fproj%2F589%2FPresentations%2FPB07-One-Sided-RCG.pdf&id=589

# Příloha 1 - Manuál

Ovládání programu je dostupné skrze uživatelské rozhraní ovládané primárně myší. V rámci programu se všechny informace ukazují na otevřených panelech. Každý panel má v sobě zvoleno jaké informace zobrazuje. Typ zobrazených informací, jde v panelu kdykoliv změnit. Pokud je v zájmu mít otevřeno více panelů, dá se tak učinit pomocí kliknutí na tlačítko *New Frame.* Pokud chcete nějaký s panelů uzavřít, každý z nich má ve svém pravém horním rohu tlačítko *Close,* které daný panel uzavře.
Tlačítka *Open Input file* a *Open grammar file* slouží k otevření vstupního textu a gramatiky. Tlačítko *Quit*, slouží k ukončení programu. Posledně zmíněná tři tlačítka jsou spolu s tlačítkem *New Frame* vždy viditelná.

Výběr, který typ informací je zobrazen v panelu, se provede pomocí jednoho z tlačítek v horní části panelu. Může se stát, že když otevřeme velké množství panelů a nebo velmi zúžíme okno programu, tlačítka na výběr informací přestanou být vidět. Jejich funkce je však stále možné dosáhnout pomocí malých šipek, které se v takovém případě zjeví v pravé horní části panelu.
V případě typů informací *Grammar* nebo *Input String*, nenabízí panel žádné další interaktivní možnosti, pouze zobrazuje daný obsah. U obsahu *LL-table* je možné označit některou z buněk tabulky, pro případné dodatkové informace. Na závěr, s možností *parsing,* se objeví uvnitř panelu dvě nová tlačítka. Tlačítko *From Start* vrátí analýzu znovu do počátečního stavu, zatímco tlačítko *Next Step* provede další krok syntaktické analýzy.

# Příloha 2 - CD