

Konverze modelů bezkontextových jazyků

Bc. David Navrkal

Diplomová práce
2016



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

nascannované zadání s. 1

nascannované zadání s. 2

ABSTRAKT

Tato práce je rozšířením mé bakalářské práce, která se zabývala, jak didakticky prezentovat studentům formálních jazyků jazyky regulární. Proto v této práci budu pokračovat jazyky bezkontextovými.

Dočtete se zde o teorii bezkontextových jazyků s množstvím příkladů. Budu se zde zabývat modely bezkontextových jazyků a to zásobníkovým automatem a bezkontextovou gramatikou. U bezkontextové gramatiky bude představena i její nejpoužívanější forma a o je Backusova-Naurova forma. Dále v teorii budou prezentovány algoritmy pro převod bezkontextové gramatiky na zásobníkový automat a zpět. Projektová část se bude zabývat návrhem uživatelského rozhraní pro převodní algoritmy a krátkou rekapitulací programu vyvinutého v rámci bp. práce do kterého tyto převodní algoritmy budu implementovat.

Klíčová slova: Bezkontextové jazyky, zásobníkový automat, bezkontextová gramatika, Backusova-Naurova forma, převody zásobníkového automatu a bezkontextové gramatiky.

ABSTRACT

This work is an extension of my bachelor thesis, which deals with how the didactically present regular languages to the students of formal languages. Therefore, this work will continue with context-free languages. You can read here about the theory of context-free languages with plenty of examples. I will discuss models of context-free languages such as pushdown automaton and context-free grammar. For context-free grammars will be presented as well its most widely used form the Backus-Naur form. Furthermore, in the theory will be presented algorithms for conversion of context-free grammar to pushdown automata and back. Project part will deal with user interface design for the conversion algorithms and a brief overview on the program developed in bachelor thesis in which I will these algorithms implement.

Keywords: Context-free languages, pushdown automata, context-free grammar, Backus-Naur form, conversions pushdown automata and context-free grammars.

poděkování, motto, úryvky knih, básní atp.

Obsah

ÚVOD	3
I TEORETICKÁ ČÁST.....	4
1 CHOMSKÉHO HIERARCHIE.....	5
2 BEZKONTEXTOVÝ JAZYK.....	6
3 ZÁSOBNÍKOVÝ AUTOMAT.....	7
4 BEZKONTEXTOVÁ GRAMATIKA	9
4.1 BACKUSOVA-NAUROVA FORMA	10
5 ALGORITMUS PŘEVODU CFG NA PDA.....	11
II IMPLEMENTACE	12
6 REFAKTOR EXISTUJÍCÍHO KÓDU	13
6.1 KONVENCE POJMENOVÁNÍ ČÁSTÍ KÓDU	13
6.2 PŘEPSÁNÍ TŘÍDY HLAVNÍHO OKNA NA NĚKOLIK MALÝCH.....	13
ZÁVĚR	15
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	17
SEZNAM OBRÁZKŮ	18

ÚVOD

Tato práce pojednává o vývoji aplikace která má didakticky prezentovat studentům teoretické informatiky oblast bezkontextových jazyků. Aplikace navazuje na bakalářskou práci, jejíž praktickou částí bylo implementovat vybrané konverze modelů bezkontextových jazyků. Toto téma jsem si vybral z toho důvodu, že oblast formálních jazyků může být pro studenty hůře pochopitelná a pomocí mé aplikace si tak mohou tuto náročnou teorii procvičit na praktických příkladech.

Zdrojové kódy a text bakalářské práce, na kterou tímto navazuji, jsou dostupné pod opensource licencí na internetovém serveru Github. Tato práce společně se zdrojovými kódy zde bude uveřejněna taktéž na <https://github.com/navrkald/regularConvertor>.

Jak již napovídá název serveru Github, práci včetně zdrojových kódů verzuji pomocí systému Git. Díky tomu mám jistotu, že ať se stane cokoliv svoje data mám tímto zálohovaná. Dále pak se mohu v případě potřeby vracet ke starším verzím. Mohu monitorovat svou práci, protože vidím kolik jsem který den udělal. Mohu také nový kód ukládat do větví, což mi umožňuje mít stabilní větev pro koncové uživatele a vývojovou větev, ve které si vyvíjím novou funkcionalitu. Jelikož jsou zdrojové kódy veřejně přístupné, tak v případě zájmu může kdokoliv přispět k v vývoji této výukové aplikace a v případě potřeby může provést takzvaný pull request, který mohu schválit a začlenit jej do aplikace.

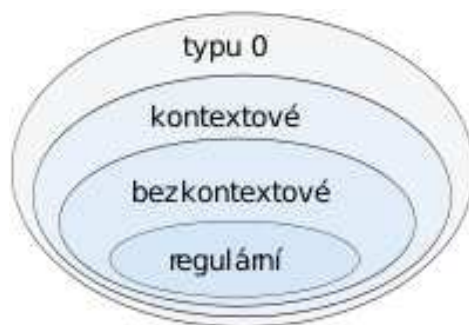
Pro vývoj tohoto programu jsem si zvolil jazyk C++, protože je mi velmi blízký, jelikož jej používám ve svém povolání programátora ve firmě AVG a také na tento jazyk byl kladen na mé předchozí škole FIT VUT v Brně. Jelikož se pohybuji ve svém životě na pomezí světa Windows a Linuxu, tak pro mě bylo zásadní, aby aplikace fungovala na těchto dvou operačních systémech, tak jsem si zvolil pro vývoj grafického uživatelského prostředí knihovnu Qt verze 5, která je v současné době nejnovější.

Tuto práci jsem se na rozdíl od své bakalářské, kterou jsem psal v programu Open Office, rozhodl psát v jazyce L^AT_EX. Toto rozhodnutí má hned několik důvodů. Prvním z nich je to, že jsem se chtěl tento jazyk už dlouho naučit, protože mi je jako programátorovi mnohem bližší. Nejen technické texty vypadají mnohem lépe v tomto jazyce, protože se za mě stará za základní typografická pravidla a také mám při psaní v L^AT_EXu větší kontrolu nad textem. Dalším důvodem, je že pomocí jazyka L^AT_EX můžu mít práci napsanu v textovém formátu, na rozdíl od programů Open Office nebo Microsoft Word, které používají binární formát, který si nemohu otevřít v poznámkovém bloku, nebo nad ním dělat jednoduše porovnání dvou verzí. Tato skutečnost znamená i mnohem efektivnější verzování jednotlivých commitů v Gitu, ve kterém si pak můžu prohlížet a porovnávat jednotlivé verze a případně se i vrátit k některé z předchozích verzí.

I. TEORETICKÁ ČÁST

1 CHOMSKÉHO HIERARCHIE

Abychom si mohli dát bezkontextové jazyky do souvislosti s ostatními formálními jazyky, je nutné si vysvětlit Chomského hierarchii. Tato hierarchie byla vytvořena panem Noamem Chomskym v roce 1956. Dává do souvislosti jazyky regulární, bezkontextové, kontextové a jazyky typu 0. Jejich vztah, můžete vidět na obrázku 1, jako vztah množin.



Obrázek 1. Chomského hierarchie formálních jazyků

Jak je vidět z obrázku regulární jazyky zaujímají nejmenší množinu, nebo chcete-li nejmenší počet jazyků. Nenechte se však zmýlit pojmem počet jazyků, protože počet jazyků a množství řetězců generovaných daným jazykem jsou dvě rozdílné věci. Například jednoduchým regulárním výrazem $(a + b)^*$ lze popsat všechny slova nad abecedou $\Sigma = a, b$. Avšak jazyk, všech slov, kde první půlka slova se skládá ze stejného počtu písmen 'a' jako písmen 'b' v druhé půlce slova regulárním jazykem nepopíšeme. Dalo by se tak říci, že čím výše se jazyk nachází v Chomského hierarchii, tím přesněji dokážeme jazyk popsat, jinými slovy, tím více jazyků jsme schopni generovat.

Nyní si pro lepší představu u každého jazyku uvedeme modely, které daný jazyk popisují s jedním příkladem konkrétního jazyka.

Regulární jazyky můžeme popsat pomocí regulárních gramatik, regulárním výrazem nebo konečným automatem. Příkladem je například jazyk který obsahuje libovolnou kombinaci písmen 'a' a 'b' končící písmenem 'b'. Tento jazyk by se dal popsat regulárním výrazem $(a + b)^*b$.

Jazyky bezkontextové lze popsat modely jakými jsou bezkontextové gramatiky a konečný zásobníkový automat. Všechny jazyky regulární jsou zároveň bezkontextové, avšak obrácené tvrzení neplatí. Příkladem je jazyk $L = \{a^n b^n : n \geq 1\}$

Jazyky kontextové můžeme popsat lineárně ohraničeným Turingovým strojem. Příkladem nechť je jazyk $L = \{a^n b^n c^n : n \geq 1\}$

Třída jazyků typu nula, v sobě obsahuje všechny doposud zmíněné jazyky plus ještě jazyky navíc. Popisujeme je úplným Turingovým strojem. [2]

2 BEZKONTEXTOVÝ JAZYK

V předchozí kapitole jsme si uvedli vzájemný vztah formálních jazyků. Nyní se pojďme podívat podrobněji na bezkontextové jazyky. Tyto jazyky popisujeme pomocí dvou hlavních modelů, kterými jsou bezkontextová gramatika (CFG) a konečný zásobníkový automat (PDA).

Nenechme se však zmýlit představou, že pomocí obou můžeme generovat bezkontextové jazyky. Protože pomocí bezkontextové gramatiky můžeme jazyk generovat, avšak pomocí konečného zásobníkového automatu můžeme rozpoznat, zda konkrétní jazyk je popsán daným automatem.

Pro nás je každopádně důležitá skutečnost, že oba popisy jsou vzájemně ekvivalentní, tj. že lze převést PDA na CFG a taktéž lze převést CFG na PDA. Formální popis bezkontextové gramatiky a zásobníkového automatu bude uveden dále spolu s jejich vzájemnými konverzemi.

Pro lepší představu jaké jsou to vlastně ty bezkontextové jazyky si nyní uvedeme pár příkladů. Krásným příkladem z praxe by mohl být jazyk závorek, tak aby odpovídal počet levých závorek počtu závorek pravých. Složitějším příkladem by mohl být jazyk matematicky správných algebraických výrazů, který se skládá z operátorů plus, mínus, krát, děleno, symbolů 'x', 'y', 'z' a závorek. Dalším příkladem z praxe je, že velká podmnožina programovacích jazyků se dá popsat pomocí bezkontextové gramatiky.

Jak je vidět z těchto příkladů, pomocí bezkontextového jazyka se dají popsat složitější věci a jejich uplatnění můžeme nalézt u popisu programovacích jazyků. Existuje notace bezkontextové gramatiky, zvaná Backusova-Naurova forma, která se právě často používá pro popis syntaxe programovacích jazyků. Znalost bezkontextových jazyků lze využít při konstrukci překladače, nebo interpretu programovacího jazyka a také při počítačovém zpracování algebraických výrazů. [3]

3 ZÁSObNÍKOVÝ AUTOMAT

Nyní si definujeme definici zásobníkového automatu. [4]

Konečný zásobníkový automat M je uspořádaná sedmice $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ kde:

- Q je konečná množina vnitřních stavů,
- Σ je konečná vstupní abeceda,
- Γ je konečná abeceda zásobníku,
- δ je přechodová relace, popisující pravidla činnosti automatu, je definovaná jako konečná množina kartézského součinu $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \times Q \times \Gamma^*$,
- q_0 je počáteční stav z množiny Q ,
- z_0 je počáteční symbol na vrcholu zásobníku z množiny Γ ,
- F je množina koncových stavů, $F \subseteq Q$.

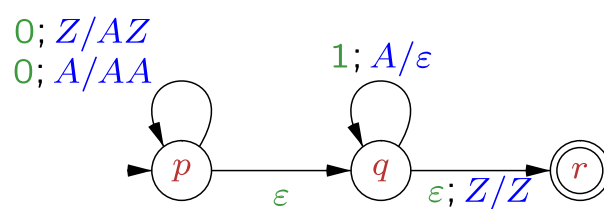
Definice 1. Definice zásobníkového automatu.

Pro lepší představu si uvedeme příklad zásobníkového automatu M , který popisuje jazyk $L = \{0^n 1^n, \text{ kde } n \geq 0\}$.

Nechť M je definován následovně:

- $Q = \{p, q, r\}$,
- $\Sigma = \{0, 1\}$,
- $\Gamma = \{A, Z\}$,
- $\delta = (p, 0, Z, p, AZ), (p, 0, A, p, AA), (p, \epsilon, Z, q, Z), (p, \epsilon, A, q, A), (q, 1, A, q, \epsilon), (q, \epsilon, Z, r, Z),$
- $q_0 = p$,
- $z_0 = Z$,
- $F = \{r\}$.

Graficky znázorněný stejný zásobníkový automat by vypadal následovně:



Obrázek 1. Příklad zásobníkového automatu

4 BEZKONTEXTOVÁ GRAMATIKA

Definice bezkontextové gramatiky je následující.

Bezkontextová gramatika G je uspořádaná čtveřice $G = (V, \Sigma, R, S)$ kde:

- V je konečná množina non-terminálních symbolů,
- Σ je konečná množina terminálních symbolů,
- R je konečná množina relací z V do $(V \cup \Sigma)^*$, taktéž se nazývá množinou přepisovacích pravidel,
- $S \in V$ se nazývá počáteční non-terminální symbol gramatiky.,

Definice 2. Definice bezkontextové gramatiky.

Jednotlivé přepisovací pravidla často píšeme pomocí „šipkové notace“, kde na levé straně neterminál za ním následuje šipka a na pravé straně je řetězec non-terminálních symbolů. Přepisovací pravidla se nazývají protože, řetězce daného jazyka, které jsou popsány gramatikou vznikají tak, že na počátku máme řetězec skládající se z počátečního non-terminálního symbolu gramatiky S a na ten aplikuji jedno z přepisovacích pravidel. Pokračuji tím, že postupně nahrazuji non-terminální symboly v řetězci do té doby, dokud mě v něm nezbudou jen terminální symboly a to je pak jedno konkrétní slovo daného bezkontextového jazyka.

Například gramatika $G = (\{S, B\}, \{a, b\}, R = \{\{S \rightarrow aBb\}, \{B \rightarrow aBb\}, \{B \rightarrow \varepsilon\}\}, S)$ popisuje jazyk $L = \{a^n b^n : n \geq 1\}$.

Příklad konkrétního slova, které generuje tato gramatika by mohl vypadat následovně:

1. Na začátku máme slovo skládající se z počátečního symbolu S .
2. Po aplikaci přepisovacího pravidla $S \rightarrow aBb$ dostaneme slovo aBb .
3. Toto pravidlo aplikujeme ještě jednou a dostáváme slovo $aaBbb$.
4. Nakonec aplikujeme pravidlo $B \rightarrow \varepsilon$ a dostáváme slovo $aa\varepsilon bb$.
5. Jelikož ε značí prázdný řetězec, tak výsledné slovo jazyka L je $aabb$. Slovo se nám skládá jen z terminálních symbolů, takže jsme s generováním slova u konce.

4.1 Backusova-Naurova forma

Pro zápis bezkontextové gramatiky se používají převážně dvě notace a to Backusova-Naurova forma (BNF) a jako druhá v pořadí je Van Wijngaardenova gramatika. My si zde představíme pouze první z nich, protože s BNF se můžete setkat častěji. Jazyky, které jsou popsány BNF a rozšířenou BNF jsou Algol, Paskal, C/C++, Ada 95, PL/I a další.

Syntaktická přepisovací pravidla jsou definována tak, že levou stranu vždy tvoří non-terminál, za kterým následuje oddělovač „::=” (dve dvojtečky následované rovnítkem) za kterým následuje pravá strana pravidla, která se skládá z řetězců non-terminálů a terminálů. Pravá strana se může skládat z více řetězců oddělených symbolem „|“ značící logický operátor OR. Tento operátor byl zaveden, kvůli optimalizaci zápisu pravidel, tak aby místo zápisu, že jeden non-terminální symbol se může rozepsat na N řetězců, což by muselo být rozepsáno na N pravidel na N řádcích. Místo toho se těchto N řetězců napíše na pravou stranu oddělených symbolem „|“. Terminální symboly jsou uvedeny v uvozovkách. [6]

Gramatika pro popis číselné konstanty by mohla vypadat takto:

```
<konstanta> ::= <číslice> | <konstanta> <číslice>  
<číslice> ::= „0“ | „1“ | „2“ | „3“ | „4“ | „5“ | „6“ | „7“ | „8“ | „9“
```

Gramatika který by popisovala základní aritmetické operace by měla navíc tato pravidla:

```
<výraz> ::= <výraz> „+“ <výraz>  
<výraz> ::= <výraz> „-“ <výraz>  
<výraz> ::= <výraz> „×“ <výraz>  
<výraz> ::= <výraz> „/“ <výraz>  
<výraz> ::= „(“ <výraz> „)“  
<výraz> ::= <konstanta>
```

5 ALGORITMUS PŘEVODU CFG NA PDA

Nyní si pojdme ukázat algoritmus převodu bezkontextové gramatiky na zásobníkový automat. [7]

Vstup: CFG $G = (V, \Sigma, R, S)$

Výstup: Nedeterministický PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$, který přijímá vstupní řetězec vyprázdněním zásobníku.

1. $Q = q_0$
2. $\Sigma = \Sigma$
3. $\Gamma = V \cup \Sigma$
4. Počáteční stav je q_0
5. Konstrukce přechodové množiny δ :
 - (a) Pro každé $a \in \Sigma$: přidej $(q_0, a, a, q_0, \varepsilon)$ do δ
 - (b) Pro každé $A \rightarrow x$, kde $A \in V, x \in (V \cup \Sigma)^*$, přidej do $\delta(q_0, \varepsilon, A, q_0, y)$, kde $y = reversal(x)$
6. $z_0 = S$
7. $G = \emptyset$

Algoritmus 1. Převod CFG na PDA.

Po aplikaci tohoto algoritmu nám vznikne nedeterministický PDA s jedním vstupním stavem a žádným ukončovacím stavem. Tento automat přijímá vstupní řetězec vyprázdněním zásobníku. Má stejnou vstupní abecedu, jako byla abeceda terminálů u původní CFG. Zásobníková abeceda automatu vznikla sloučením množin terminálů a non-terminálů vstupní gramatiky. Počátečním symbolem na zásobníku je počáteční non-terminální symbol gramatiky.

První část množiny přechodových pravidel vznikla tak, že pro všechny terminální symboly se přidala přechodová pravidla taková, že počáteční i koncový stav pravidla je vždy počáteční stav automatu. Zároveň aktuální symbol na pásce a stejně tak na vrcholu zásobníku musí být stejný terminální symbol. Znak ε říká, že symbol z vrcholu zásobníku odstraním.

Zbytek přechodových má taktéž počáteční a koncový stav stejný. ε zde říká, že nečtu žádný ze symbolů na pásce, ale přečtu jeden non-terminál z vrcholu zásobníku a nahradím jej řetězcem znaků z pravé části pravidla napsaných pozpátku.

II. IMPLEMENTACE

6 REFAKTOR EXISTUJÍCÍHO KÓDU

Refaktoring je činnost při, které se zlepšuje kvalita kódu bez přímého vlivu na uživatele aplikace.

Nerefaktorovaný kód vede k tomu, že jen obtížně mohou programátoři rozumět kódu a to vede k častým chybám a prodloužením vývoje.

Kód by měl být pokud možno homogení a měl by v něm být dodržován systém pojmenovávání.

6.1 Konvence pojmenování částí kódu

Pro pojmenovávání proměnných, tříd a typů jsem použil systém používaný ve firmě AVG. Vše popisuje následující tabulka.

konvence	popis	poznámka
m_<název členské proměnné>	Název členské proměnné.	Písmeno <i>m</i> podle anglického slova <i>member</i>
C_<Název třídy>	Název třídy.	Písmeno <i>C</i> Podle anglického slova <i>class</i>
I_<Název rozhraní>	Název rozhraní.	Písmeno <i>I</i> podle anglického slova <i>interface</i>

Tabulka 1. Konvence pro pojmenovávání tříd a proměnných

Dalšími konvencemi jsou pojmenovávání pomocí "CamelCase". To znamená, že pokud se nějaký název skládá z více slov, tak jsou jednotlivá slova od sebe graficky oddělena pomocí toho, že každé další slovo je od přechodného odděleno velkým písmenem.

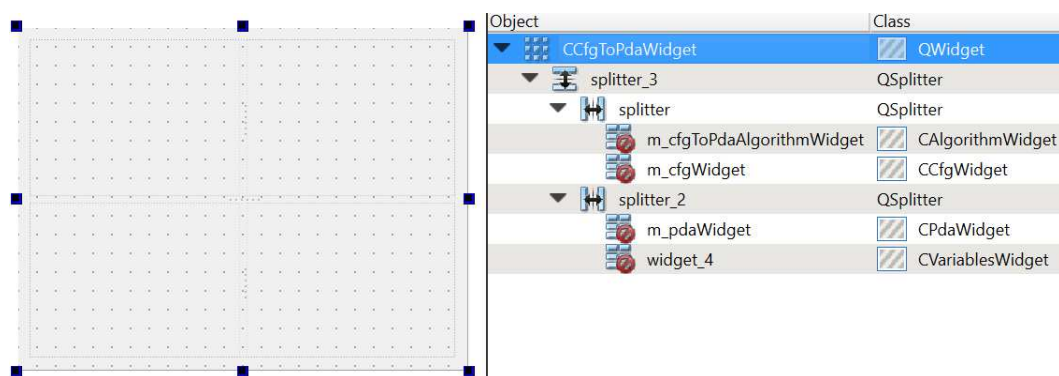
Při pojmenovávání souborů jsem se snažil dodržet konvenci, že název souboru se skládá pouze z malých písmen následovaný příponou souboru.

6.2 Přepsání třídy hlavního okna na několik malých

Příliš velké metody nebo třídy nejsou tolik přehledné a udržitelné. Proto jsem se rozhodl přepsat moji třídy hlavního okna do více modulů, chcete-li tříd.

V mém původním návrhu bakalářské práce jsem neuměl techniky zvané *Promoting*. Je to technika v Qt Designeru, kde si navrhnu vzhled widgetu, tak že se skládá, podobně jako stavebnice z ostatních uživatelských widgetů. Toto je možné, protože každý grafický prvek musí mít za předka QWidget.

Na obrázku můžete vidět widget pro konverzi



Obrázek 1. Příklad zásobníkového automatu

ZÁVĚR

V této práci jsem se snažil k hůře stravitelné teorii přikládat množství příkladů, aby byla tato problematika pochopitelná a přínosná pro studenta, který by si tuto práci chtěl půjčit v univerzitní knihovně. Z vlastní zkušenosti vím, že pro lepší pochopení dané problematiky je kromě suché teorie nutné přiložit i příklady. Dával jsem je zde i pro to, že konečným cílem diplomové práce bude kromě textu nutné vytvořit i funkční aplikaci, která má sloužit především studentům, ale samozřejmě ji mohou využít i profesori na naší škole k prezentaci přednášené látky pro lepší přiblížení studentům. Chtěl jsem, aby i student nepříliš seznámený s danou problematikou mohl po přečtení této práce začít používat mou aplikaci bez obtíží.

SEZNAM POUŽITÝCH ZDROJŮ

- [1] KOPKA, Helmut, Patrick W DALY a Jan GREGOR. *Latex: podrobný průvodce*. Vyd. 1. Brno: Computer Press, 2004, 576 s. ISBN 80-7226-973-9.
- [2] *Chomsky hierarchy* [online]. [cit. 2015-11-16]. Dostupný z WWW: https://en.wikipedia.org/wiki/Chomsky_hierarchy.
- [3] *Context-free language* [online]. [cit. 2015-11-17]. Dostupný z WWW: https://en.m.wikipedia.org/wiki/Context-free_language.
- [4] *Pushdown automaton* [online]. [cit. 2015-11-17]. Dostupný z WWW: https://en.wikipedia.org/wiki/Pushdown_automaton.
- [5] *Context-free grammar* [online]. [cit. 2015-11-17]. Dostupný z WWW: https://en.m.wikipedia.org/wiki/Context-free_grammar.
- [6] *Backus-Naur Form* [online]. [cit. 2015-11-18]. Dostupný z WWW: https://en.wikipedia.org/wiki/Backus-Naur_Form.
- [7] HOPCROFT, John E, Rajeev MOTWANI a Jeffrey D ULLMAN. *Introduction to automata theory, languages, and computation*. 2nd ed. Boston: Addison-Wesley, 2001, xiv, 521 s. ISBN 0201441241.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

CFG Context-free grammar, česky taktéž bezkontextová gramatika

PDA Pushdown automaton, česky též zásobníkový automat

BNF Backusova-Naurova forma. (Notace pro zápis bezkontextové gramatiky.)

Seznam obrázků

Obr. 1. Chomského hierarchie formálních jazyků	5
Obr. 1. Příklad zásobníkového automatu	8
Obr. 1. Příklad zásobníkového automatu	14