Diplomová práce

Kateřina Sloupová

2020

Název práce česky	Reimplementace a konsolidace systémů podporujících výuku formálních jazyků
Název práce anglicky	Reimplementation and Consolidation of Systems for Formal Languages Educatio
Klíčová slova	formální jazyky, automatické vyhodnocování, Python
Autor	Bc. et Bc. Kateřina Sloupová
Vedoucí	doc. RNDr. Jan Strejček, Ph.D.
Konzultant	RNDr. Vladimír Štill

Na tomto místě se v tištěné práci nachází oficiální podepsané zadání práce a prohlášení autora školního díla.

Prohlášení

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

Kateřina Sloupová

Abstrakt

Poděkování

Obsah

1 Úvod **1**

2 Konečné reprezentace formálních jazyků **3**
2.1 Formální jazyky 3
2.2 Deterministické konečné automaty 4
2.3 Nedeterministické konečné automaty (NFA) 7
2.4 Regulární gramatiky 8
2.5 Regulární výrazy 8

3 Implementace **9**
3.1 Deterministické konečné automaty 9
3.2 Nedeterministické konečné automaty 9
3.3 Regulární gramatiky 9
3.4 Regulární výrazy 9
3.5 Programovací jazyky 10
3.6 Testy 10
3.7 Parsování 10

4 Vyhodnocování **11**
4.1 Přehled souvisejících služeb 11

5 Shrnutí **15**

Příloha **17**

Kapitola 1

Úvod

Na Fakultě informatiky Masarykovy univerzity již od jejího založení *ověř* probíhá výuka formálních jazyků a automatů, neboť jejich pochopení studentům pomáhá např. při práci s regulárními výrazy, parsery nebo kompilátory. Konkrétně jde o dva předměty s obdobnou strukturou IB005 Formální jazyky a automaty [1] a IB102 Automaty a gramatiky[2] (pouze do roku 2019; v letech 2012–2016 vypisován ve verzi rozšířené o základy složitosti pod názvem Automaty, gramatiky, složitost). Na oba navazuje předmět IA006 Vybrané kapitoly z teorie automatů[3].

Zhruba od roku 2009 je používán při výuce vyhodnocovací nástroj pro rozhodnutelné problémy z oblasti formálních jazyků. Za tuto dobu se na dvacet závěrečných prací studentů FI MU (viz přílohu 5), především pod vedením doc. RNDr. Jana Strejčka, Ph.D., věnovalo jeho vývoji, zdokonalování, rozšíření o generátory příkladů k řešení a také integraci s tzv. odpovědníky Informačního systému Masarykovy univerzity (dále IS), skrze něž probíhá samotný e-learning. Aktuálně jsou odpovědníky v rámci IS používány k procvičování a zkoušení v rámci domácích úkolů, s vkládáním gramatik, automatů a regulárních výrazů pomáhá editor s automatickou kontrolou syntaxe a následně je uvedeným vyhodnocovacím nástrojem ověřena správnost vyřešení úlohy.

Zdroj některých problémů při využívání a úpravách vyhodnocovacích služeb tkví v nemožnosti upravit samotné vyhodnocovací jádro z roku 2009, jde totiž o zkompilovaný javovský program bez dostupných zdrojových kódů. Po vícero pracích se záměrem napravit působené problémy nyní přistupujeme ke komplexnímu řešení, a to k nové implementaci jádra vyhodnocovací služby, na něž lze navázat již fungující ostatní nástroje. Ziskem bude mimo jiné zařa-

[1] https://is.muni.cz/auth/predmet/fi/IB005
[2] https://is.muni.cz/auth/predmet/fi/IB102
[3] https://is.muni.cz/auth/predmet/fi/IA006

zení námětů ke zlepšení, získaných za roky používání vyhodnocovací služby při výuce, přizpůsobení současným potřebám výuky, použití jazyka umožňujícího jednodušší úpravy a dostatečná dokumentace.

Kapitola 2

Konečné reprezentace formálních jazyků

Definice formalismů v této kapitole jsou ve stručnosti převzaty ze studijních materiálů předmětů IB005 Formální jazyky a automaty a IB102 Automaty a gramatiky, konkrétně ze skript k oběma předmětům[?] a prezentací a přednášek k předmětu IB102.

2.1 Formální jazyky

Teorie popsaná v následujících sekcích slouží k popisu formálních jazyků obsahujících až nekonečné množství slov konečnými prostředky.

Abeceda Σ je konečná množina znaků (písmen, symbolů), např. $\{a, b\}$, $\{0, 1, \ldots, 9\}$ nebo $\emptyset$.

Slovo v nad abecedou Σ je libovolná konečná posloupnost znaků této abecedy, např. řetězec *aaba* je slovo nad abecedou $\{a, b\}$. Prázdné slovo neboli posloupnost s nulovým počtem znaků se značí jako ε. Slova lze řetězit za sebe (i vícenásobně – operace iterace). *Bude-li třeba pro další text, doplnit.*

Jazyk L je libovolná množina slov nad abecedou Σ, např. nad abecedou $\{a, b\}$ existují jazyky jako $\{aa, bb, ab, ba\}$, $\{a, aa, aaa, aaaa, \ldots\}$ nebo $\emptyset$. Speciálně množinu všech slov nad abecedou Σ značíme jako Σ^* (* je operace iterace jazyka, jejímž výsledkem je nový jazyk všech možných kombinací řetězení slov z původního jazyka). Z definice iterace také platí $\varepsilon \in \Sigma^*$ pro libovolnou abecedu Σ.

Klasifikace formálních jazyků

Chomského hierarchie

Regulární jazyky Tato práce se zabývá pouze teorií a implementací algoritmů pro třídu regulárních jazyků.

Příklad *Uvidím, co použiju v definicích a algoritmech, a doplním – ať nepřepisuju celý skripta.*

2.2 Deterministické konečné automaty

Deterministický konečný automat

Deterministický konečný automat (*deterministic finite automaton*, DFA) [1] $\mathcal{M}$ je uspořádaná pětice $(N, \Sigma, \delta, q_0, F)$, kde

- N je konečná neprázdná množina stavů,
- Σ je konečná množina znaků, řečená abeceda,
- δ je přechodová funkce typu $N \times \Sigma \to N$,
- q_0 je počáteční stav, $q_0 \in N$, a
- F je množina koncových (akceptujících) stavů, $F \subseteq N$.

Jazyk akceptovaný deterministickým konečným automatem je množina všech jím akceptovaných slov – takových, při jejichž čtení automat dokáže přejít z iniciálního stavu do některého z koncových stavů. Formálně definujme nejprve rozšířenou přechodovou funkci $\hat{\delta} : N \times \Sigma^* \to N$ induktivně:

$$\hat{\delta}(q, \varepsilon) = q \text{ pro } \forall q \in N$$

$$\hat{\delta}(q, wa) = \begin{cases} \delta(r, a) & \text{je-li } \hat{\delta}(q, w) = r \text{ i } \delta(r, a) \text{ definováno,} \\ \perp & \text{jinak,} \end{cases}$$

kde a je znak a w slovo nad abecedou Σ, symbol $\perp$ označuje nedefinovaný výsledek funkce. Formálně je tedy jazyk $L(\mathcal{M})$ přijímaný automatem $\mathcal{M}$:

$$L(\mathcal{M}) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}$$

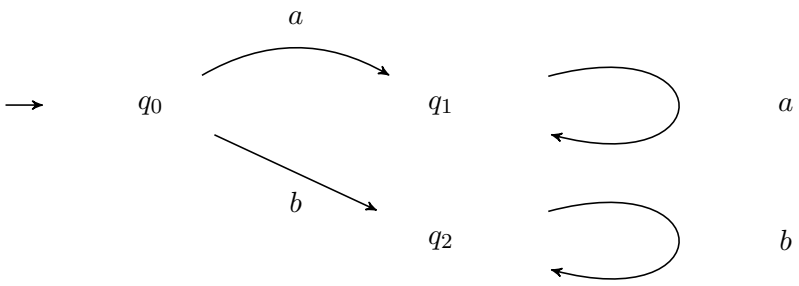
Příklad Následuje příklad deterministického konečného automatu: $\mathcal{M} = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_1, q_2\})$, kde

$$\delta(q_0, a) = q_1 \qquad \delta(q_1, a) = q_1$$
$$\delta(q_0, b) = q_2 \qquad \delta(q_2, b) = q_2$$

[1] Popis implementace DFA je v sekci 3.1.

Konečný automat lze zapsat také jako tabulku přechodové funkce nebo přechodový graf: *(graf spravím, mám špatně tikz)*

	a	b
$\rightarrow q_0$	q_1	q_2
$\leftarrow q_1$	q_1	$-$
$\leftarrow q_2$	$-$	q_2



Jazyk přijímaný tímto automatem bude zahrnovat slova o délce minimálně jednoho znaku právě taková, která obsahují buď pouze znaky a, nebo pouze znaky b, tedy:

$$L(\mathcal{M}) = \{x^n \mid x \in \Sigma, n \geq 1\}$$

Totální DFA

Ke každému deterministickému konečnému automatu $\mathcal{M}$ existuje ekvivalentní totální DFA $\mathcal{M}'$ s totální přechodovou funkcí (všechny přechody jsou definované). Sestrojit jej lze přidáním nového nekoncového stavu p a tam, kde v přechodové funkci přechod chybí, přidat nový přechod do p. *(přidat formálně? asi jo)*

Příklad Automat $\mathcal{M}$ z předchozího příkladu není totální, ekvivalentní totální DFA $\mathcal{M}'$ vytvoříme přidáním stavu p a chybějících přechodů:

	a	b
$\rightarrow q_0$	q_1	q_2
$\leftarrow q_1$	q_1	p
$\leftarrow q_2$	p	q_2
p	p	p

Minimální DFA

Definice

Algoritmus

Příklad Automat užitý jako předešlý příklad už je minimální. Můžeme například uvést automat akceptující stejný jazyk a minimalizovat jej *ne, dej sem jinej.* Mějme automat $\mathcal{N}$ zadaný tabulkou:

	a	b
$\rightarrow q_0$	q_1	q_2
$\leftarrow q_1$	q_1	p
$\leftarrow q_2$	p	q_2
p	p	p

Kanonický DFA

Kanonizace automatu spočívá v přejmenování stavů určeným způsobem. Akceptují-li dva automaty stejný jazyk, pak musí být po provedení minimalizace a kanonizace obou automatů shodné.

V kontextu této práce a vyhodnocovací služby volíme pojmenování velkými písmeny anglické abecedy, je-li třeba více než 26 stavů, zacházíme s písmeny jako s číslicemi v poziční číselné soustavě[2]:

A, B, ..., Y, Z, AA, AB, ..., AZ, BA, BB, ..., ZY, ZZ, AAA, AAB, ...

pořadí pojmenovávání - algoritmus?

Příklad

	a	b
$\rightarrow A$	B	C
$\leftarrow B$	B	D
$\leftarrow C$	D	C
D	D	D

Paralelní synchronní kompozice automatů, doplněk

algoritmus

Příklad

Ekvivalence automatů

algoritmus

Převod DFA na NFA

Nedeterministický konečný automat, definovaný v sekci 2.3, se liší od deterministického pouze typem přechodové funkce, která je namísto $N \times \Sigma \rightarrow N$

[2] https://en.wikipedia.org/wiki/Bijective_numeration#The_bijective_base-26_system

typu $N \times \Sigma \to N^2$, tedy přechod ze stavu pod daným znakem vede do množiny stavů místo jednoho stavu. Pro převod DFA $\mathcal{M}$ s přechodovou funkcí δ na NFA $\mathcal{N}$ a δ' ji stačí jednoduše upravit přidáním množin:

$$\delta'(q,a) = \begin{cases} \{\delta(q,a)\} & \text{je-li } \delta(q,a) \text{ definováno,} \\ \bot & \text{jinak,} \end{cases}$$

2.3 Nedeterministické konečné automaty (NFA)

Nedeterministický konečný automat

nondeterministic finite automaton, NFA, je to samý, jenom typ přechodové funkce je $N \times \Sigma \to N^2$.

Příklad

Determinizace NFA

Ke každému nedeterministickému konečnému automatu lze vytvořit ekvivalentní deterministický automat. Princip determinizace spočívá ve tvorbě nových stavů odpovídajících množinám stavů, do nichž vedou přechody v automatu.

Algoritmus

Příklad

Nedeterministický konečný automat s ε-kroky

NFA s ε-kroky vznikne rozšířením přechodové funkce NFA o možnost přechodu do nového stavu nejen pod znakem abecedy, ale také pod prázdným slovem, tedy typ přechodové funkce bude $N \times \Sigma \cup \varepsilon \to N^2$.

Odstranění ε-kroků

algoritmus

Převod NFA na regulární gramatiku

algoritmus

Převod NFA na regulární výraz

algoritmus

2.4 Regulární gramatiky

Regulární gramatika

Regulární gramatika[3] je uspořádaná čtveřice (N, Σ, P, S), kde

- N je konečná neprázdná množina neterminálů,
- Σ je konečná množina terminálů,
- P je množina pravidel tvaru $A \to a$ nebo $A \to aB$, $A, B \in N, a \in \Sigma$ a
- S je počáteční neterminál, $S \in N$.

V množině pravidel P je povoleno také pravidlo $S \to \varepsilon$ v případě, že S je počáteční neterminál a nevyskytuje se v pravé straně žádného z pravidel.

Převod regulární gramatiky na NFA

algoritmus

2.5 Regulární výrazy

Regulární výraz

Množina regulárních výrazů [4] nad abecedou Σ je definována induktivně:

1. ε, $\emptyset$ a a pro všechna $a \in \Sigma$ jsou (základní) regulární výrazy nad Σ;
2. jsou-li E, F regulární výrazy nad Σ, jsou jimi i $(E.F), (E + F)$ a (E^*);
3. každý regulární výraz vznikne konečným počtem aplikací kroků 1 a 2.

V regulárních výrazech se mohou vyskytovat také závorky určující rozsah operátorů. Nejvyšší prioritu má operátor $*$, poté . a nejmenší $+$, nadbytečné závorky lze vypouštět.

Regulární výraz E nad abecedou Σ určuje jazyk $L(E)$ následovně:

$$\begin{aligned} L(\varepsilon) &= \{\varepsilon\} & L(E.F) &= L(E).L(F) \\ L(\emptyset) &= \emptyset & L(E+F) &= L(E) \cup L(F) \\ L(a) &= \{a\}, a \in \Sigma & L(E^*) &= L(E)^* \end{aligned}$$

Převod regulárního výrazu na NFA

algoritmus

[3] Popis implementace regulárních gramatik je v sekci 3.3.
[4] Popis implementace regulárních výrazů je v sekci 3.4.

Kapitola 3

Implementace

Odevzdaná implementace se nachází v archivu závěrečné práce[1], aktuální dále vyvíjená verze je v repozitáři *zatím na gitlabu, ale neveřejném.*

Grafika formalismů a toho, co umí

3.1 Deterministické konečné automaty

Třída DFA je implementována jako pětice splňující požadavky kladené definicí [2].
`DFA: Tuple[Set[State]], Set[Character], ...`

3.2 Nedeterministické konečné automaty

3.3 Regulární gramatiky

[3]

3.4 Regulární výrazy

[4]

[1] https://is.muni.cz/auth/th/bwci5/
[2] DFA jsou definovány v sekci 2.2.
[3] Regulární gramatiky jsou definovány v sekci 2.4.
[4] Regulární výrazy jsou definovány v sekci 2.5.

3.5 Programovací jazyky

Užívaný Python:
Mypy, verze, volby
pep

3.6 Testy

přehled, přidávání testů

3.7 Parsování

Gramatika a generování parseru

Tvorba parserů pomocí ANTLR4: https://github.com/antlr/antlr4
Gramatiky pro parsery podle Poklemby:https://is.muni.cz/auth/th/h6afo/

CLASSPATH=".:/usr/local/lib/antlr-4.8-complete.jar:$CLASSPATH"
alias antlr4='java -Xmx500M -cp "/usr/local/lib/antlr-4.8-complete.jar:$CLASSPATH" org.antlr.v4.Tool'
pip install antlr4-python3-runtime

Použití vygenerovaného parseru v implementaci

Parser, Lexer, Listener, Visitor

Kapitola 4

Vyhodnocování

4.1 Přehled souvisejících služeb

Vyhodnocovací nástroj

text

Informační systém Masarykovy univerzity

Informační systém Masarykovy univerzity (dále jen IS) je provozován a vyvíjen od roku 1999 a je používán nejen všemi fakultami univerzity, ale i dalšími školami a univerzitami. Představuje společný rámec pro agendy související se studiem (organizace, administrativa, komunikace, e-learning) a mnohé další. Více informací o IS a jeho funkcích a možnostech je k dispozici na jeho webových stránkách [1].

Odpovědníky

text

Editor pro vkládání

text

Generátory příkladů

text

hsExprTest

text

[1] https://is.muni.cz/nas_system/

Use cases v ekosystému vyhodnocovací služby

Vyučující připravuje/opravuje úlohu pro zkoušku. *Vygeneruje si ji generátorem.* Převedení formalismu požadovaným způsobem či jeho kontrolu může provést na webové stránce vyhodnocovací služby.

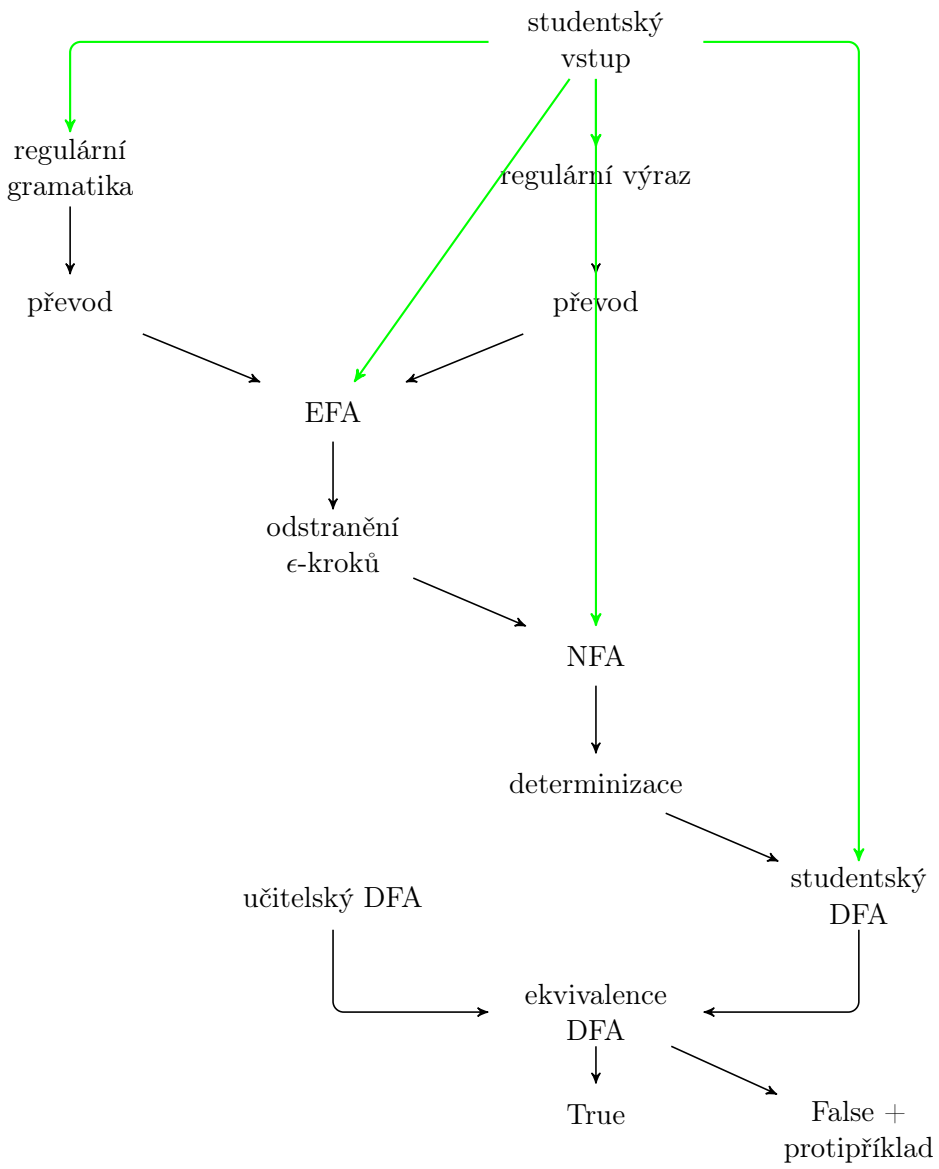
Vyučující zadává domácí úkol v předmětu. *text*

Student plní domácí úkol. *text*

Student se připravuje na zkoušku. Kromě dostupných procvičovacích odpovědníků v IS, které fungují podobně jako ty pro domácí úlohy, lze studovat z dostupných učebních materiálů k předmětu. Potřebuje-li student pro kontrolu porovnat dva formalismy nebo převést formalismus určitým způsobem, může využít webovou službu.

Realizované testování provozu vyhodnocovací služby v IS

domácí úkoly, procvičování, zkoušky



Obrázek 4.1: Schéma úloh.

Kapitola 5

Shrnutí

Příloha

Přehled souvisejících bakalářských a diplomových prací

2008

Automatická kontrola syntaxe obsahu textové oblasti ve webovém formuláři

2009

Výuková pomůcka pro předmět Automaty a gramatiky
Internetová služba pro transformace bezkontextových gramatik
Generátor príkladov k teórii formálnych jazykov
Generátor příkladů k teorii formálních jazyků

2011

Pokročilá internetová služba pro řešení rozhodnutelných problémů z oblasti regulárních jazyků
Rozšiřitelný nástroj pro práci s formálními jazyky

2012

Generátor príkladov k teórii formálnych jazykov

2013

Implementace LR analyzátorů v prostředí JGAF
LL analyzátory a jejich implementace v prostředí JGAF
Pokročilá internetová služba pro řešení rozhodnutelných problémů z oblasti bezkontextových jazyků

2014

Pohodlné vkládání regulárních jazyků do webového formuláře
Rozšíření nástroje JGAF
Pokročilá internetová služba pro řešení rozhodnutelných problémů z oblasti

formálních jazyků

2015

Generátor příkladů k teorii formálních jazyků
Konsolidace nástroje JGAF
Pohodlné vkládání regulárních jazyků do webového formuláře

2016

Pohodlné vkládání regulárních jazyků do webového formuláře

2018

Pokročilá podpora pro odpovědníky z oblasti formálních jazyků

Houdek

JGAF