VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DVOUDIMENSIONÁLNÍ KONEČNÉ AUTOMATY A JEJICH APLIKACE

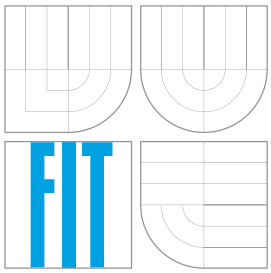
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ZDENĚK HLADÍK

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

DVOUDIMENSIONÁLNÍ KONEČNÉ AUTOMATY A JEJICH APLIKACE

TWO-DIMENTIONAL FINITE AUTOMATA AND THEIR APPLICATIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ZDENĚK HLADÍK
AUTHOR

VEDOUCÍ PRÁCE — prof. RNDr. ALEXANDER MEDUNA, CSc.
SUPERVISOR

BRNO 2015

Abstrakt

Tato práce představuje teorii dvoudimensionálních jazyků a automatů v porovnání s klasickými jazyky a automaty. Této teorie je využito k vytvoření speciálního automatu, který umožňuje analýzu tabulek obsahujících pravidla. Tyto tabulky jsou spojeny s pravidly registračních značek vybraných evropských států. Automat a tabulky jsou používány vyvinutou aplikací, která umožňuje identifikaci státní příslušnosti vozidel.

Abstract

This paper introduces the theory of twodimensional languages automata in comparision to the classic concept of languages and automata. This theory is used for developing special automata, that is capable of analyzing tables containing rules. These tables are conected with rules for selected european car license plates. Automata and tables are used by developed application, which can be used for car nationality identification.

Klíčová slova

dvoudimensionální automat, dvoudimensionální jazyky, registrační značky, identifikace státní příslušnosti, státní příslušnost vozidla

Keywords

twodimensional automata, twodimensional languages, license plates, number plates, nationality identification, vehicle nationality

Citace

Zdeněk Hladík: Dvoudimensionální konečné automaty a jejich aplikace, bakalářská práce, Brno, FIT VUT v Brně, 2015

Dvoudimensionální konečné automaty a jejich aplikace

Prohlášení

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

.....
Zdeněk Hladík
14. května 2015

Poděkování

Chtěl bych poděkovat vedoucímu této práce, prof. RNDr. Alexanderu Medunovi, CSc., za jeho čas a spolupráci.

Obsah

1 Úvod **3**

2 Jazyky a konečné automaty **4**
2.1 Základní pojmy . . . 4
2.1.1 Abecedy a řetězce . . . 4
2.1.2 Jazyky . . . 5
2.2 Regulární jazyky . . . 5
2.3 Konečný automat . . . 6

3 Dvoudimensionální jazyky **8**
3.1 Jazyk ve více dimenzích . . . 8
3.2 Dvoudimensionální jazyky . . . 8
3.2.1 Obrazy a abecedy . . . 9
3.2.2 Jazyk . . . 11
3.3 Regulární výrazy . . . 12

4 Konečné automaty pracující s obrazy **13**
4.1 Dvoudimensionální automaty . . . 13
4.1.1 Zástupci . . . 13
4.1.2 Použití . . . 15
4.2 Vyvíjený automat . . . 16
4.2.1 Tabulka pravidel . . . 16
4.2.2 Sloupcový dvoudimensionální automat . . . 17
4.2.3 Automat jako struktura . . . 18
4.2.4 Čtyřsměrný automat . . . 18
4.2.5 Třísměrný automat . . . 19

5 Registrační značky **22**
5.1 RZ v Evropě . . . 22
5.1.1 Štítky se zkratkou státu . . . 22
5.2 Pravidla ve státech . . . 23
5.2.1 Česká republika . . . 23
5.2.2 Slovenská republika . . . 24
5.2.3 Polsko . . . 25
5.2.4 Německo . . . 26
5.2.5 Rakousko . . . 26

6 Aplikace **28**
6.1 Aplikace pro vyhledávání . . . 28
6.1.1 Existující řešení . . . 28
6.1.2 Databáze aplikace . . . 31
6.1.3 Program . . . 33
6.1.4 Grafické uživatelské rozhraní . . . 36
6.1.5 Příklady použití a testování . . . 37
6.1.6 Shrnutí . . . 39
6.2 Aplikace pro editaci databáze RZ . . . 39
6.2.1 Program . . . 40
6.2.2 Shrnutí a popis použití aplikace . . . 42

7 Závěr **44**

A Obsah CD **48**

B Vzory registračních značek **49**
B.1 Česká republika . . . 49
B.2 Slovensko . . . 50
B.3 Polsko . . . 50
B.4 Německo . . . 51
B.5 Rakousko . . . 51
B.6 Belgie . . . 52
B.7 Velká Británie . . . 52
B.8 Řecko . . . 53
B.9 Nizozemsko . . . 53
B.10 Ruská federace . . . 54
B.11 Turecko . . . 54

Kapitola 1

Úvod

Tato práce se zabývá zasazením teorie dvoudimensionálních jazyků do praktického využití. Konkrétně se jedná o vývoj aplikace, jenž umožňuje uživateli určit státní příslušnost zadané registrační značky vozidla.

V práci jsou představeny základní termíny teorie formálních jazyků s důrazem na vysvětlení principu fungování konečných automatů a jim podobných prostředků, jenž umožňují určit náležitost řetězců do příslušných formálních jazyků. Tato látka je jednou z nejzákladnějších v oboru teoretické informatiky.

Dále je dříve představená teorie zasazena do dvou dimenzí, tedy jsou představeny obrazy, dvoudimensionální ekvivalenty řetězců, a jazyky těchto obrazů. Jsou demonstrovány nové operace, které mohou být prováděny nad dvoudimensinálními jazyky. Tyto operace jsou později použity v praxi. Také je představeno několik možností, jakými lze poznatky z oblasti vícedimensionálních jazyků využít, jako je třídimensionální reprezentace těles a materiálů.

V souvislosti s teorií dvoudimensionálních jazyků a jejich formálními zástupci jsou představeny i regulární výrazy a konečné automaty, umožňující určit náležitost obrazů do těchto jazyků. Dvoudimensionální konečné automaty jsou představeny ve formě dvou zástupců, na kterých je demonstrován postup automatu při analýze obrazu.

Pro účel aplikace vyvíjené v rámci této práce bylo nutné zkonstruovat originální automat, jenž bude analyzovat tzv. *tabulky pravidel*, které obsahují pravidla pro řetězce daného jazyka. Tyto pravidla představují formáty registračních značek vozidel a jazyky těchto pravidel jsou množiny pravidel příslušných států. Analýza tabulek pak představuje vyhledávání jazyků, které podporují zadanou konfiguraci automatu, tedy registrační značku.

V práci jsou představeny formáty registračních značek několika zemí a je určeno, které státy a tedy i registrační značky bude aplikace v době dokončení práce podporovat. Konkrétní státy, které budou detailně analyzovány, jsou Česká republika a sousední státy. Databáze aplikace by však měla obsahovat veškeré státy, jejichž vozidla se významně podílejí na provozu v České republice.

Jako poslední je představena aplikace a její databáze registračních značek. Jsou analyzovány jiné práce a aplikace zabývající se tímto tématem a provedeno zhodnocení jejich přínosu pro společné využití s touto aplikací. Je popsána struktura databáze a postup, který aplikace používá při navigaci v této databázi během vyhledávání odpovídajících pravidel pro zadanou registrační značku. Následuje testování a shrnutí získaných výsledků.

Kapitola 2

Jazyky a konečné automaty

Tato kapitola obsahuje definici a vysvětlení pojmů vztahujících se k teorii konečných automatů. Tyto pojmy jsou nezbytné k pochopení problematiky a jsou použity i v kapitolách 2 a 3 při představení dvoudimensionálních jazyků a konečných automatů. Tato látka je náplní předmětu IFJ (Formální jazyky a překladače), proto je čerpáno z přednášek tohoto předmětu a knihy garanta předmětu prof. Alexandera Meduny [12].

2.1 Základní pojmy

Tato podkapitola zahrnuje definice těch nejzákladnějších pojmů, jako je abeceda, řetězec a jazyk.

2.1.1 Abecedy a řetězce

Definice 1. *Abeceda* je konečná, neprázdná množina elementů, které nazýváme symboly.

Abeceda je často označována symbolem Σ.

Definice 2. Nechť Σ je abeceda.

1. ε je řetězec nad Σ
2. Pokud x je řetězec nad Σ a $a \in \Sigma$, potom xa je řetězec nad abecedou Σ.

V definici 2 je ε prázdný řetězec.

Řetězec nad abecedou je tedy libovolnou posloupností symbolů abecedy, přičemž délka řetězce je počet jeho symbolů. Tedy ε má délku 0.

Dále uvádím dvě důležité operace pro práci s řetězci: konkatenace a mocnina řetězce.

Definice 3. Nechť x a y jsou dva řetězce nad abecedou Σ. *Konkatenace* x a y je řetězec xy.

Výsledkem konkatenace řetězce s prázdným řetězcem je původní řetězec.

Definice 4. Nechť x je řetězec nad abecedou Σ. Pro $i \geq 0$,i-tá *mocnina* řetězce x, značíme x^i, je definována:

1. $x^0 = \varepsilon$
2. pro $i \geq 1 : x^i = xx^{i-1}$

2.1.2 Jazyky

Jazyk je množina slov (řetězců), které dávají v kontextu jazyka nějaký smysl. Porovnáme-li například češtinu a angličtinu, abecedy řečí jsou podobné, každý jazyk ale obsahuje jinou množinu slov, která jej zároveň i definuje.

Definice 5. Nechť Σ^* značí množinu všech řetězců nad Σ. Každá podmnožina L $\subseteq \Sigma^*$ je *jazyk* nad Σ.

Stejně jako řetězec jazyk podporuje i některé operace, které jsou nutné k jejich značení regulárními výrazy. Pro nás důležité operace jsou: konkatenace, iterace a mocnina jazyka. Zároveň má stejně jako množina vlastnost konečnosti či nekonečnosti. Následují definice těchto termínů.

Definice 6. Nechť L_1 a L_2 jsou dva jazyky nad Σ. *Konkatenace jazyků* L_1 a L_2 je definována jako: $L_1L_2 = \{xy : x \in L_1 \text{ a } y \in L_2\}$

Definice 7. Nechť L je jazyk nad abecedou Σ. *Iterace jazyka* L, značíme L^*, a *pozitivní iterace jazyka* L, značíme L^+, jsou definovány následovně: $L^* = \bigcup_{i=0}^{\infty} L^i$, $L^+ = \bigcup_{i=1}^{\infty} L^i$

Definice 8. Nechť L je jazyk nad abecedou Σ. Pro $i \geq 0$, i-tá *mocnina jazyka* L, značíme L_i, je definována:

1. $L^0 = \{\varepsilon\}$
2. pro $i \geq 1 : L^i = LL^{i-1}$

Definice 9. Jazyk L je *konečný*, pokud L obsahuje konečný počet řetězců, jinak je *nekonečný*.

2.2 Regulární jazyky

Pokud konečný jazyk obsahuje přesný počet řetězců, lze pro něj vytvořit příslušnou gramatiku, tedy množinu pravidel, jenž splňují pouze obsažené řetězce viz str. 210 v [12]. Obecně platí, že každý konečný jazyk je *regulární* (definovatelný regulárními pravidly) viz teorém 3.1.1 v [12]. Pravidla jsou, jak už bylo zmíněno, často zapisována formou regulárních výrazů, proto následuje definice regulárních výrazů a jazyků.

Definice 10. Nechť Σ je abeceda. *Regulární výrazy* nad abecedou Σ a *jazyky*, které *značí*, jsou definovány následovně:

$\oslash$ je regulární výraz značící prázdnou množinu
ε je regulární výraz značící jazyk $\{\varepsilon\}$
a, kde $a \in \Sigma$, je regulární výraz značící jazyk $\{a\}$
Nechť r a s jsou regulární výrazy značící po řadě jazyky L_r a L_s, potom:

- $(r.s)$ je regulární výraz značící jazyk $L = L_rL_s$
- $(r + s)$ je regulární výraz značící jazyk $L = L_r \cup L_s$
- (r^*) je regulární výraz značící jazyk $L = L_r^*$

Definice 11. Nechť L je jazyk nad abecedou Σ. L je *regulární jazyk* nad Σ, pokud je popsán regulárním výrazem r nad Σ.

2.3 Konečný automat

Konečný automat je jednoduchý výpočetní stroj, který se může nacházet v jednom stavu z definované množiny stavů, přechody mezi těmito stavy jsou ovlivněny vstupy, které automat přijímá. Následuje definice konečného automatu.

Definice 12. *Konečný automat* je pětice: $M = (Q, \Sigma, R, s, F)$, kde

Q je *konečná množina stavů*
Σ je *vstupní abeceda*
R je *konečná množina pravidel* tvaru: $pa \to q$, kde $p, q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$
$s \in Q$ je *počáteční stav*
$F \subseteq Q$ je množina *koncových stavů*

Chování konkrétního konečného automatu pro konkrétní vstup je závislé na jeho konfiguraci a přechodů na základě těchto konfigurací, tyto vlastnosti určují, jak se změní stav konečného automatu nacházejícího se v určitém stavu při příjmu konkrétního vstupu. Následují definice konfigurace, přechodu a sekvence přechodů v konečném automatu.

Definice 13. Nechť $M = (Q, \Sigma, R, s, F)$ je konečný automat. *Konfigurace* konečného automatu M je řetězec $\chi \in Q\Sigma^*$

Definice 14. Nechť pax a qx jsou dvě konfigurace konečného automatu M, kde $p, q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$ a $x \in \Sigma^*$. Nechť $r = pa \to q \in R$ je pravidlo. Potom M může provést *přechod* z pax do qx za použití r, zapsáno $pax \vdash qx[r]$ nebo zjednodušeně $pax \vdash qx$

Definice 15. Nechť χ_0, χ_1, ..., χ_n je sekvence přechodů konfigurací pro $n \geq 1$ a $\chi_{i-1} \vdash \chi_i[r_i]$, $r_i \in R$ pro všechna $i = 1, \ldots, n$, což znamená: $\chi_0 \vdash \chi_1[r_1] \vdash \chi_2[r_2] \ldots \vdash \chi_n[r_n]$. Pak M provede *n-přechodů* z chi_0 do chi_n; zapisujeme: $\chi_0 \vdash^n \chi_n[r_1 \ldots r_n]$ nebo zjednodušeně $\chi_0 \vdash^n \chi_n$.

Pomocí přechodů a stavů lze v konečném automatu detekovat, zda přijatý vstup splňuje nějaká pravidla, což ve spojení s předchozí teorií značí, že konečný automat lze využít k určení, zda přijímané posloupnosti znaků, tedy řetězce, splňují gramatiku jazyka, tedy zda je přijatý řetězec řetězcem nad konkrétním jazykem. Jazyk, jehož gramatika odpovídá konfiguraci konečného automatu, je *jazyk přijímaný automatem.*

Definice 16. Nechť $M = (Q, \Sigma, R, s, F)$ je konečný automat. *Jazyk přijímaný* konečným automatem M, $L(M)$, je definován: $L(M) = \{w : w \in \Sigma^*, sw \vdash^* f, f \in F\}$

Ovšem je reálná možnost, že vstup není žádným z řetězců přijímaného jazyka, v tomto případě je nutné tuto skutečnost ihned po jejím zjištění indikovat přechodem do stavu, který značí nepříslušnost vstupu do jazyka. Není žádoucí, aby se při nedefinovaném vstupu automat "zasekl", tedy dosáhl *neukončujícího stavu*, což je stav, pro nějž neexistuje řetězec pro přechod do koncového stavu, a také není žádoucí, aby pro nějaký vstup neexistoval přesně 1 přechod, přičemž neexistuje přechod pro prázdný vstup, tedy aby KA byl *deterministický.* Takovýto automat se nazývá *úplný deterministický konečný automat*, viz následující definice.

Definice 17. Nechť $M = (Q, \Sigma, R, s, F)$ je *deterministický konečný automat. M* je *úplný*, pokud pro libovolné $p \in Q$, $a \in \Sigma$ existuje právě jedno pravidlo $pa \to q \in R$ pro nějaké $q \in Q$. Jinak je automat neúplný.

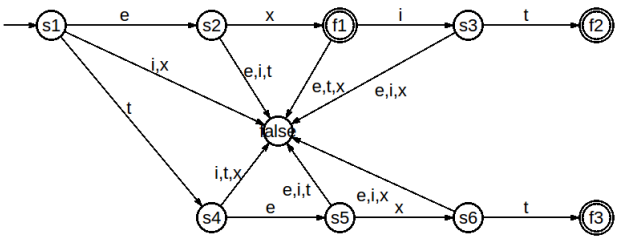
Mluvíme-li o úplném DKA s pouze jedním neukončujícím stavem (ten indikuje poruchu) a žádným stavem nedostupným, nazývá se tento typ automatu *dobře specifikovaný KA*.

Každý konečný automat může být znázorněn pomocí tabulky přechodů, ve které je pro každý stav uveden stav následující při odpovídajícím znaku na vstupu automatu. Jiná, názornější možnost vizualizace automatu je jeho grafický model, kde jsou zaneseny stavy a přechody mezi nimi pomocí bodů a šipek s popisky.

Příklad 1. Pro demonstraci uvedených pojmů uvádíme jednoduchý příklad automatu $M1 = (Q, \Sigma, R, s, F)$, kde

$Q = \{s1, s2, \ldots, s6, f1, f2, f3\}$
$\Sigma = \{e, i, x, t\}$
R je znázorněno tabulkou 2.1
$s = s1$
$F = \{f1, f2, f3\}$

Automat M1 přijímá jazyk $L = \{ex.exit, text\}$ a je znázorněn obrázkem 2.1.



Obrázek 2.1: Konečný automat M1

	$\delta_{\mathbf{M1}}$	**x**	**i**	**t**	**e**
→	s1	false	false	s4	s2
	s2	f1	false	false	false
	s4	false	false	false	s5
	false				
←	f1	false	s3	false	false
	s5	s6	false	false	false
	s3	false	false	f2	false
	s6	false	false	f3	false
←	f2				
←	f3				

Tabulka 2.1: Tabulka přechodů pro M1

V této kapitole byla představena a demonstrována teorie vztahující se k základní podobě konečných automatů, další kapitola je věnována komplexnějším automatům pro vícerozměrné gramatiky. Teorie v této kapitole je uvedena pouze v nezbytném rozměru, proto odkazujeme na zmíněnou literaturu pro širší pochopení problematiky.

Kapitola 3

Dvoudimensionální jazyky

Tato kapitola se zabývá již definovanou látkou v oblasti dvou dimenzí. Tímto rozšířením je umožněn popis složitějších vzorů a jejich klasifikace pomocí automatů. V bakalářském programu Fakulty informačních technologií nejsou tyto pojmy přednášeny, proto je čerpáno převážně z knihy [14] a práce [4].

3.1 Jazyk ve více dimenzích

Dříve definovaný pojem jazyka lze zařadit i do více dimenzí, kdy už není množinou řetězců, ale vícerozměrnýnch vzorů. Tyto vzory mají podobné vlastnosti jako jednorozměrné řetězce, lze je spojovat či provádět jiné operace pro jejich úpravu. Také podléhají nějaké abecedě, z jejichž symbolů jsou vytvořeny.

Pokud se omezíme na nejvíce tři dimenze, může být třídimensionální řetězec jakýmsi tělesem se standardními rozměry, tedy šířkou, výškou a hloubku, složeným ze symbolů odpovídající abecedy. Takto lze teoreticky popsat libovolné těleso. Pokud vytvoříme abecedu $\Sigma = \{0, 1\}$, kde 0 znamená prázdný prostor a 1 znamená látku tělesa, je vícerozměrný řetězec těleso v prostoru.

Pokud do tohoto scénáře zapojíme jazyky, lze například vytvořit nekonečný jazyk obsahující všechny krychle a pro tento jazyk vytvořit regulární výraz i konečný automat, který bude procházet prostorem a látkou tělesa a zkoumat, zda je těleso krychle.

Takováto myšlenka je dnes při populárním trendu třírozměrného tisku velmi zajímavá, například pro kontrolu kvality výrobků či formální klasifikaci těles. Mimo abecedy obsahující znaky pro přítomnost materiálu by bylo možné navrnout například abecedy obsahující prvky značící barevný odstín nebo druh materiálu. V praxi je tato teorie, i když nepřímo, používána ve všech modelářských nástrojích.

Více než třírozměrné jazyky a gramatiky jsou složitější pro člověka na vizualizaci, nicméně v teoretickém popisu není pro ně překážky, avšak tato teorie je příliš komplexní a v nynější době stále neprozkoumána.

3.2 Dvoudimensionální jazyky

Dvoudimensionální jazyky jsou stejně jako dříve představené klasické jazyky jednodimensionální na nižší úrovni oproti zmíněným vícedimesionálním jazykům. Tyto jazyky mohou být rozeznávány pomocí stavových automatů a zároveň existují formální prostředky pro popis dvoudimensionálních vzorů, neboli *obrazů*. Na rozdíl od zmíněné komplexnosti složitějších

jazyků jsou dvoudimensionální jazyky dostatečně jednoduché, aby teorie, která se k nim vztahuje, mohla být popsána a pochopena. Stále jsou však dostatečně pokročilé natolik, aby nám oproti klasickým jazykům umožňovaly více možností při jejich klasifikaci.

Vztah mezi 2–d jazyky a 1–d jazyky lze demonstrovat následovně: Jak bylo zmíněno, dvoudimensionální jazyky jsou množinou obrazů, což jsou pokročilé ekvivalenty řetězců. Pokud se omezíme na obrazy o velikosti (viz dále) (1,n) nebo (n,1) nejsou tyto obrazy ničím jiným než řetězci. Z toho vyplývá, že teorii užívanou pro dvoudimensionální jazyky lze aplikovat i na jednodimensionální řetězce. Tento poznatek je použit i v aplikaci vyvinuté v rámci této práce.

3.2.1 Obrazy a abecedy

Následují definice termínů odpovídajících termínům v kapitole 2, přičemž stále platí tvrzení, že Σ značí abecedu. Definice a příklady jsou převzaty z knihy [14]. Kompletní formální definice jednotlivých operací a jejich demonstrace viz [7].

Definice 18. *Dvoudimensionální řetězec* (nebo *obraz*) nad abecedou Σ je dvoudimensionální obdélníkové pole skládající se z elementů abecedy Σ.

Pro demonstraci: tabulka 3.1 znázorňuje obraz P1 nad abecedou $\Sigma = \{0, 1\}$.

1	0	0	0
0	1	0	0
1	0	0	0
0	1	1	0

Tabulka 3.1: Obraz P1

Definice 19. Množina všech dvoudimensionálních řetězců nad Σ je značena Σ^{**}.

Pro daný obraz $p \in \Sigma^{**}$, nechť $\ell_1(p)$ značí počet řádků obrazu p a $\ell_2(p)$ značí počet sloupců obrazu p. Dvojice $(\ell_1(p), \ell_2(p))$ je pak nazývána velikost obrazu p. *Prázdný obraz* je jediný obraz velikosti (0,0) a je poté značen λ (pozn.: analogie prázdného řetězce ε). Obrazy velikosti $(n,0)$ a $(0,n)$ nejsou definovány.

Množina všech obrazů nad Σ velikosti (m, n), kde $m, n > 0$ je značena $\Sigma^{m \times n}$.

Dále, pokud $1 \leq i \leq \ell_1(p)$ a $1 \leq j \leq \ell_2(p)$, nechť $p(i, j)$, či $p_{i,j}$ značí symbol v obrazu p se souřadnicemi (i, j).

Definice 20. Nechť p je obraz velikosti (m, n). *Blok* obrazu p je obraz p', který je podmnožinou pole p. Pokud (m', n') je velikostí obrazu p', pak $m' \leq m$ a $n' \leq n$ a existují celá čísla h,k ($h \leq m - m', k \leq n - n'$) taková, že $p'(i, j) = p(i + h, j + k)$ pro všechna $0 \leq i \leq m'$ a $0 \leq j \leq n'$.

Obraz P2 znázorněný tabulkou 3.2 je blokem obrazu P1 v tabulce 3.1

1	0	0
0	1	0

Tabulka 3.2: Obraz P2

Často se při zobrazení obrazu definují znaky, které značí jeho okraje, tyto znaky nejsou obsaženy v abecedě, nad níž je postaven obraz, pro příklad volíme znak ♯ a předchozí obraz P2.

♯	♯	♯	♯	♯
♯	1	0	0	♯
♯	0	1	0	♯
♯	♯	♯	♯	♯

Tabulka 3.3: Obraz P2 s hranicemi

Pro některé úkony s obrazy je nutné využít operace projekce, která podle zadaného klíče mapuje jeden obraz na druhý, lze ji použít k vyjádření souvislosti mezi obrazy nebo transformacím obrazů na jiné. Pro následující definici nechť Γ a Σ jsou dvě konečné abecedy a $\pi : \Gamma \to \Sigma$ je mapování, dále odkazované jako projekce.

Definice 21. Nechť $p \in \Gamma^{**}$ je obraz. *Projekce mapování* π obrazu p je obraz $p' \in \Sigma^{**}$ takový, že $p'(i,j) = \pi(p(i,j))$, pro každé $1 \leq i \leq \ell_1(p)$ a $1 \leq j \leq \ell_2(p)$.

Pro dvoudimensionální řetězce existují speciální operace: *sloupcová* a *řádková konkatenace*. Za účelem demonstrace definujeme dva čtvercové obrazy P3 a P4 nad abecedou $\Sigma = \{0, 1\}$.

1	0
0	1

Tabulka 3.4: Obraz P3

1	1
0	1

Tabulka 3.5: Obraz P4

Definice 22. *Řádková konkatenace* obrazů P3 a P4 (značena $P3 \ominus P4$) je dílčí operace, definována pouze pro $\ell_1(P3) = \ell_1(P4)$. Výsledek konkatenace je zobrazen tabulkou 3.6

Definice 23. *Sloupcová konkatenace* obrazů P3 a P4 (značena $P3 \odot P4$) je dílčí operace, definována pouze pro $\ell_2(P3) = \ell_2(P4)$. Výsledek konkatenace je zobrazen tabulkou 3.7

1	0
0	1
1	1
0	1

Tabulka 3.6: $P3 \ominus P4$

1	0	1	1
0	1	0	1

Tabulka 3.7: $P3 \odot P4$

Stejně jako konkatenace řetězců je konkatenace obrazů jednoduchou operací, již si lze snadno představit. Oproti jednodimensionálním řetězcům disponují obrazy navíc operací rotace, tato operace je logicky stejně jednoduchá na vizualizaci.

Definice 24. Nechť P2 je obraz. Rotace (ve směru ručiček) obrazu P2, značená jako $P2^R$, je definována v tabulce 3.8.

0	1
1	0
0	0

Tabulka 3.8: Obraz $P2^R$

3.2.2 Jazyk

Dvoudimensionální jazyk je stejně jako obraz pouhým zasazením předchozích pojmů do dimenze navíc, proto je definice následující.

Definice 25. *Dvoudimensionální jazyk* nad abecedou Σ je podmnožinou Σ^{**}.

Logicky se pro dvoudimensionální jazyk vyskytují i operace řádkové a sloupcové konkatenace. Tyto operace mají s příslušnými konkatenacemi obrazů stejný vztah jako konkatenace jednodimensionálních jazyků s konkatenací řetězců.

Definice 26. Nechť L_1, L_2 jsou dvoudimensionální jazyky nad abecedou Σ, pak *řádková konkatenace* L_1 a L_2 (značeno $L_1 \ominus L_2$) je definována následovně: $L_1 \ominus L_2 = \{p \ominus q | p \in L_1$ a$q \in L_2\}$

Definice 27. Nechť L_1, L_2 jsou dvoudimensionální jazyky nad abecedou Σ, pak *sloupcová konkatenace* L_1 a L_2 (značeno $L_1 \odot L_2$) je definována následovně: $L_1 \odot L_2 = \{p \odot q | p \in L_1$ a $q \in L_2\}$

Rotace jazyka L je pak rotací všech obrazů, jež jazyk L obsahuje, a značíme ji L^R. Dále pro jazyky existuje i ekvivalent projekce.

Definice 28. Nechť $L \subseteq \Gamma^{**}$ je jazyk obrazů. *Projekce mapováním* π jazyka L je jazyk $L' = \{p' | p' = \pi(p) \forall p \in L\} \subseteq \Sigma^{**}$

Pro dvoudimensionální jazyky se operace iterace nazývají *řádkový* a *sloupcový uzávěr* jazyka, jsou značeny následovně.

Definice 29. Nechť L je jazyk. *Řádkový uzávěr* jazyka L, značen $L^(* \ominus)$ je definován jako $L^(* \ominus) = \bigcup_{i \geq 0}^{\infty} L^{i\ominus}$, kde $L^{0\ominus} = \lambda$, $L^{1\ominus} = L$, $L^{n\ominus} = L \ominus L^{(n-1)\ominus}$.

Definice 30. Nechť L je jazyk. *Sloupcový uzávěr* jazyka L, značen $L^(* \odot)$ je definován jako $L^(* \odot) = \bigcup_{i \geq 0}^{\infty} L^{i\odot}$, kde $L^{0\odot} = \lambda$, $L^{1\odot} = L$, $L^{n\odot} = L \odot L^{(n-1)\odot}$.

Jak bylo řečeno, jednodimensionální a dvoudimensionální jazyky jsou v jistém měřítku kompatibilní, pro demostraci slouží následující operátor, jenž ze dvou jednodimensionálních jazyků vytvoří dvoudimensionální jazyk.

Definice 31. Nechť Σ je konečná abeceda, a nechť S_1, $S_2 \subseteq \Sigma^*$ jsou dva řetězcové jazyky nad Σ. Pak *řádková a sloupcová kombinace* řetězců S_1 a S_2 je dvoudimensionální obraz $L = S_1 \oplus S_2 \subseteq \Sigma^{**}$ takový, že obraz $p \in \Sigma^{**}$ náleží do jazyka L, pouze když řádky obrazu odpovídají jednotlivým řetězcům a sloupce obrazu p odpovídají zároveň obsahu řetězců v S_1 a S_2.

Pokud uvažujeme například řetězcový jazyk S nad abecedou Σ, který obsahuje pouze řetězce začínající znakem a, pak každý obraz s prvním sloupcem složeným pouze ze znaků a je obsažen v jazyce $L = S \oplus \Sigma^*$.

3.3 Regulární výrazy

Pomocí definovaných operací lze nad dvoudimensionálními jazyky opět vytvářet regulární výrazy pro jejich popis. Pro začátek stanovíme abecedu Σ, prázdný jazyk $\emptyset$ a každý jazyk $L = \{a\}$, kde $a \in \Sigma$, tyto jazyky se nazývají *atomické jazyky* nad Σ. Poté deklarujeme $\Re$, což je následující množina operací: $\Re = \{\ominus, \odot, *\ominus, *\odot, \cup, \cap, ^c\}$.

Elementy množiny $\Re$ nazýváme *regulární operace*. Jazyk nad abecedou Σ je regulární, pokud je složen z atomických jazyků pomocí konečného množství těchto regulárních operací. Stejně jako u jednodimensionálních jazyků *regulární výraz* je popis, jak je určitý regulární jazyk složen pomocí regulárních operací z atomických jazyků.

Definice 32. *Regulární výraz* nad abecedou Σ je definován rekurzivně jako následující:

1. $\emptyset$ a každý symbol $a \in \Sigma$ jsou regulárními výrazy
2. pokud α a β jsou regulární výrazy, pak $(\alpha) \cup (\beta)$, $(\alpha) \cap (\beta)$, $^c(\alpha)$,$(\alpha) \odot (\beta)$, $(\alpha) \ominus (\beta)$, $(\alpha)^{*\odot}$, $(\alpha)^{*\ominus}$ jsou regulární výrazy.

Definice 33. Dvoudimensionální jazyk $L \subseteq \Sigma^{**}$ je *regulární*, pokud může být značen regulárním výrazem nad abecedou Σ.

Pro demostraci síly regulárních výrazů jsou uvedeny následující příklady. Je na místě podotknout, že regulární výraz může popisovat jakýkoliv obrazec ve dvoudimensionální mřížce, tedy i standardní geometrické obrazce, jako je čtverec. Při velké abstrakci a rozsáhlé abecedě by teoreticky bylo možné popsat regulárním výrazem i tvář konkrétního člověka či otisk jeho prstu. Tento poznatek spadá spíše do oboru počítačové grafiky, je však logické, že principy podobné dvoudimensionálním regulárním výrazům se v jisté obměně používají právě pro popis vzorů v obrazu i v praxi.

Příklad 2. Nechť $\Sigma = \{a, b\}$. Pak regulární výraz $(((a \ominus b)^{*\ominus}) \odot ((b \ominus a)^{*\ominus}))^{*\odot}$ značí jazyk skládající se ze všech "šachovnicových" obrazů se sudou délkou stran. Příkladem je obraz P5.

a	b	a	b	a	b	a	b
b	a	b	a	b	a	b	a
a	b	a	b	a	b	a	b
b	a	b	a	b	a	b	a

Tabulka 3.9: Obraz P5

Kapitola 4

Konečné automaty pracující s obrazy

Tato kapitola si klade za cíl seznámit čtenáře se základní teorií dvoudimensionálních konečných automatů a představit obory, ve kterých by tyto v základě jednoduché stroje mohly být prakticky využity a to velmi efektivně. Funkce představených zástupců dvoudimensionálních automatů je demonstrována na příkladu.

Dále představujeme vývoj vlastního automatu založeného na zmíněných automatech a operacích, spolu s příklady, které vystihují klady a zápory jednotlivých variant, tyto příklady jsou demonstrovány na tvaru dat, jenž vznikl úpravou pravidel zmíněných v kapitole 5. Je představena konečná verze automatu, který je použit v aplikaci, jíž se zabývá kapitola 6.

4.1 Dvoudimensionální automaty

Dvoudimensionální konečné automaty jsou stručně stroje pohybující se po dvoudimensionální pásce (tedy případně obrazu) pomocí konečné množiny směrů na základě vnitřních pravidel. Podle počtu těchto směrů lze klasifikovat tyto automaty na čtyřsměrné, dvousměrné atd. Pro praktické použití je zapotřebí minimálního počtu dvou směrů (každého v jiné dimenzi), aby se automat mohl přesunout z počáteční polohy do jakéhokoli bodu na pásce (pohyby ve směru osy `x` a `y`).

Dále, stejně jako klasický konečný automat viz 2.3, nabývá tento automat vždy jednoho z konečné množiny stavů, jimiž lze indikovat vlastnost analyzované pásky. Touto vlastností je často příslušnost pásky do množiny (jazyka). Takovou množinou například může být množina obrazů (fotografií) daného předmětu, či lokality. Další příklady využití viz sekce 4.1.2.

4.1.1 Zástupci

První práce zabývající se tématem automatu na dvoudimensionální pásce [4] představila automat pohybující se po čtvercovém obrazu ve čtyřech směrech (nahoru, dolů, vlevo, vpravo). Nutnou vlastností analyzovaného obrazu je, že musí obsahovat i elementy, představující jeho hranice viz sekce 3.2.1.

Takový automat (začínající na pozici $(1, 1)$ v obraze) dokázal detekovat výskyt pouze jednoho druhu elementů v celém obraze, nebo například výskyt čtvercového vzoru uprostřed obrazu. V práci je mimo jiné zmíněno, že problémy, jenž může řešit klasický jednoduchý konečný automat, mohou být neřešitelné dvoudimensionálním automatem.

Čtyřsměrný automat

Jako nejlepšího zástupce pro demonstraci principů fungování těcho automatů volíme *čtyřsměrný automat*, představený v [14] a [9], kde je uvedeno, že nedeterministická verze tohoto automatu je mnohem mocnější než automat z práce [4], jelikož dokáže analyzovat několik tříd dvoudimensiálních jazyků. Jeho definice je následující:

Definice 34. *Čtyřsměrný automat*, označován jako $4DFA$ je sedmice $\Lambda = (\Sigma, Q, \Delta, q_0, q_a, q_r, \delta)$, kde:

Σ je vstupní abeceda;
Q je konečná množina stavů;
$\Delta = \{R, L, U, D\}$ je množina "směrů"(right,left,up,down);
$q_0 \in Q$ je počáteční stav;
$q_a, q_r \in Q$ jsou "akceptující" a "odmítající" stavy;
$\delta : Q \setminus \{q_a, q_r\} \times \Sigma \to Q \times \Delta$ je přechodová funkce.

Automat pak začíná analyzovat obraz od pozice $(1, 1)$ se stavem q_0, pohybuje se po obrazu ve směrech z množiny Δ po obraze, měnící svůj stav podle δ. Funkce δ rozhoduje o směru pohybu a změně stavu automatu tak, že při přechodu do stavu q_a je určeno, že přijímaný obraz spadá do příslušného jazyka, naopak přechod do stavu q_r značí, že obraz do jazyka nepatří.

Teselační automat

Dalším představitelem zajímavých automatů je dvousměrný on-line teselační automat (představený v [8]), jenž využívá teorie *celulárních automatů* (na FIT VUTBR vyučováno v předmětu Modelování a simulace), což jsou ve zkratce pole buněk, z nichž každá nabývá jednoho stavu v každém čase. Změny těchto stavů jsou ovlivněny stavy buněk okolních a předchozím stavem buňky.

Tento automat je vyjímečný tou vlastností, že se nepohybuje po vstupní pásce, analýza je provedena přechody stavů buněk odpovídajícího *celulárního automatu*. Definice automatu je následující:

Definice 35. *Dvoudimensionální on-line teselační automat*, značíme $2 - DOTA$, je definován jako $A = (\Sigma, Q, I, F, \delta)$, kde:

Σ je vstupní abeceda;
Q je konečná množina stavů;
$I \subseteq Q$ je množina počátečních stavů;
$F \subseteq Q$ je množina konečných stavů;
$\delta : Q \times Q \times \Sigma \to Q$ je přechodová funkce.

Analýza pak začíná v čase $t = 0$, kdy je stav $q_0 \in I$ vložen do všech pozic v prvním řádku a sloupci analyzovaného obrazu p o velikosti (i, j), v následujících časových úsecích jsou měněny stavy buněk, z nichž každá odpovídá jednomu elementu obrazu, na základě stavů dvou buněk sousedních, buňky nad současnou buňkou a buňkou vlevo od současné buňky, a znakem na pozici obrazu, zaspisujeme funkcí $\delta(q_1, q_2, p(i, j))$. Po ustálení stavů v celém poli buněk, je-li stav buňky na pozici $p(i, j)$ stavem z množiny F, pak je obraz přijat automatem.

4.1.2 Použití

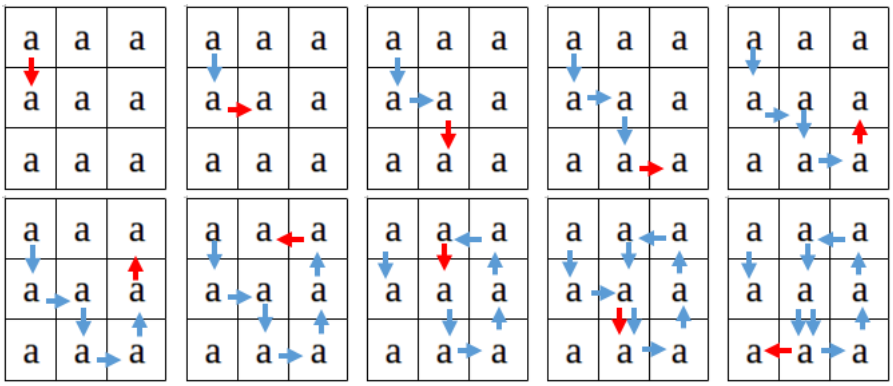
Pro demonstraci použití definujeme jazyk L nad abecedou $\Sigma = \{a\}$ takový, že obsahuje pouze čtvercové obrazy, formálně $L = \{p | p \in \Sigma^{**} \quad a \quad \ell_1(p) = \ell_2(p)\}$. Také definujeme obraz p_1 zobrazený tabulkou 4.1, který bude automaty analyzován.

a	a	a
a	a	a
a	a	a

Tabulka 4.1: Obraz p_1

Příklad 3. Automat $4 - DFA$ dokáže určit náležitost vstupního obrazu do jazyka L. Počáteční pozice automatu je $(1, 1)$, poté se automat pohybuje diagonálně (tedy střídavě ve směrech L, D), pokud automat tímto způsobem dosáhne pravého spodního rohu obrazu, provede analýzu všech elementů obrazu, zda odpovídají znaku a. Pokud jsou oba tyto kroky úspěšné, je obraz přijat automatem a náleží do jazyka L.

Obrázek 4.1 demonstruje postup automatu obrazem.



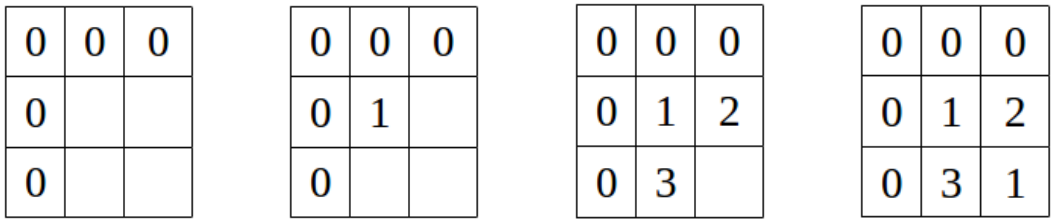
Obrázek 4.1: Postup automatu 4DFA

Příklad 4. Automat $2-DOTA$ určí náležitost obrazu do jazyka L při následující kofiguraci:

$Q = \{0, 1, 2, 3\}$
$I = \{0\}$
$F = \{1\}$
$\delta(0, 0, a) = \delta(2, 3, a) = 1$
$\delta(0, 1, a) = \delta(0, 2, a) = \delta(2, 1, a) = \delta(2, 2, a) = 2$
$\delta(1, 0, a) = \delta(3, 0, a) = \delta(1, 3, a) = \delta(3, 3, a) = 3$

Stavy pole buněk příslušného automatu v jednotlivých časech jsou zachyceny v obrázku 4.2. Z obrázku je zřejmý postup změn stavů buněk, zároveň v čase $t = 3$ se v pravém dolním rohu nachází buňka se stavem 1, což je konečný stav, jazyk je tedy přijmut automatem.

Na obrázku lze vidět, že automat postupuje mnohem rychleji než čtyřsměrný automat, proto by byl vhodnější volbou pro implementaci. Naopak by ovšem byl implementačně složitější a jeho konstrukce pro zadaný příklad je mnohem náročnější.



Obrázek 4.2: Postup automatu 2-DOTA

Dvoudimensionální automaty lze použít (jak už bylo řečeno) pro rozeznávání obrazových, či textových polí. Ve vyšší formě abstrakce takové použití zahrnuje například rozeznávání typů objektů.

Pokud bychom sestrojili jazyk obsahující všechny snímky objektu, například automobilu, pak by bylo možné pro tento jazyk vyvinout automat, jenž by procházel vstupní snímky a detekoval, zda jde o snímky daného automobilu.

Stejného principu je možné využít v oblasti biometrického zabezpečení, teoreticky lze sestrojit automat odpovídající jazyku, který obsahuje obrazy reprezentující model tváře uživatele. takový automat pak rozezná podle vlastních postupů, zda je vstupní snímek obrazem uživatelovy tváře. Ekvivaletně lze automatů využít i pro zabezpečení otiskem prstu, či speciálním obrazovým klíčem (např. v podobě QR kódu).

Neposlední využití automatů by zahrnovalo popisy obrysů horských masivů, či krajinných horizontů, které by opět mohly být reprezentovány ve formě jazyků. Výsledný konečný automat by poté určoval, zda fotografie zachycuje danou oblast, či by umožnil detekovat, která oblast se na snímku nachází.

4.2 Vyvíjený automat

Ze začátku je nutné podotknout, že vyvíjený automat je postaven na teorii dvoudimensionálních automatů, ale jeho vstupy jsou odlišné. Účelem automatu je vyhledat v dvoudimensionálním poli, které obsahuje pravidla, pravidlo takové, které pokrývá zadaný řetězec. Takovým zadaným řetězcem je tvar registrační značky analyzovaného automobilu a tabulka je uložena ve formě souboru v databázi vyvinuté aplikace.

Pro pochopení vyvinutého automatu jako automatu dvoudimensionálního je nutné si činnost automatu vyložit tak, že při načtení vstupního řetězce r je abstraktně vytvořen jazyk, který obsahuje všechny obrazy p takové, že $\ell_1(p) = |r|$ a zároveň p obsahuje sloupec reprezentující pravidlo, které odpovídá r. Poté je činností automatu analýza všech tabulek v databázi s $\ell_1(p) = |r|$ a detekce sloupců s odpovídajícími pravidly. Jde tedy o vyhledávání sloupců v tabulkách, které potvrdí jejich náležitost do vytvořeného jazyka. Výstupem použití automatu na databázi aplikace pak je množina všech tabulek přijatých automatem.

4.2.1 Tabulka pravidel

Forma tabulky vznikla pomocí operací dvoudimensionálních jazyků, pro každý stát jsou vytvořeny tabulky s pravidly jednotlivých délek. Tyto pravidla jsou uložena ve formě řetězců, které byly pomocí *řádkové konkatenace* spojeny v tabulku, na kterou je následovně

aplikována operace *rotace* kvůli minimalizaci výpočetní náročnosti. Vstupní soubory totiž pak mohou být načítány po jednotlivých řádcích.

Pro demonstraci algoritmů jednotlivých automatů bude využito vzoru registrační značky `BO123E`, což je registrační značka českého automobilu, určeného pro export (viz 5.2.1).

Odpovídající soubor s pravidly je pak zobrazen na obrázku 4.3 (pouze část souboru). Jsou přítomny zástupné znaky pro minimalizaci obsahu souborů, kde: ⋆ značí libovolný znak, $ značí libovolné písmeno a ♯ značí libovolnou číslici.

```
A B B B B C C C C C C C D J J J J J P P P P Z Z *
$ E I O K B E H K L R V C C E H I N C Z E H N R *
# # # # # # # # # # # # # # # # # # # # # # # *
# # # # # # # # # # # # # # # # # # # # # # # *
# # # # # # # # # # # # # # # # # # # # # # # *
# # # # # # # # # # # # # # # # # # # # # # # E
```

Obrázek 4.3: Tabulka pravidel

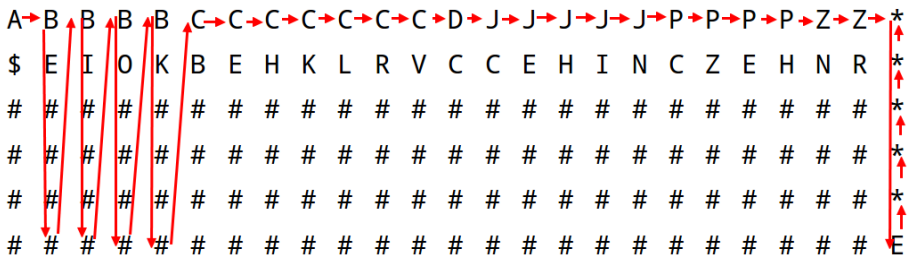
Následuje výčet a popis verzí automatů, jenž byly navrhovány za účelem použití v aplikaci.

4.2.2 Sloupcový dvoudimensionální automat

Tento automat vychází z fungování čtyřsměrného automatu viz 4.1.1. Jeho vstupy jsou tabulka pravidel a analyzovaný řetězec. Využívá čtyř směrů pohybu po tabulce pravidel a jeho stavy a pohyb po tabulce pravidel jsou závislé na tvaru vstupního řetězce a elementu na aktuálních souřadnicích. Automat začíná analyzovat tabulku pravidel od souřadnic $(1, 1)$.

Přechodová funkce automatu je sestavena tak, aby automat procházel jednotlivé sloupce tabulky pravidel a analyzoval, zda jejich první prvek obsahuje element odpovídající prvnímu znaku v řetězci. Při nesplnění této podmínky pokračuje automat ve směru R. Pokud je nalezen odpovídající element, je proveden přechod na konec tohoto sloupce, kde je opět testován element na shodu s posledním znakem řetězce. Při neúspěchu je pokračováno ve směru $\ell_2(p) \times U$ a R, tedy se provede návrat na první řádek, při úspěchu jsou analyzovány elementy uvnitř sloupce směrem U. Konec vyhledávání je určen přechodem automatu mimo tabulku (vyhledávání neuspěšné), nebo navštívením již analyzovaného elementu (úspěšné vyhledávání).

Demonstrace pohybu automatu po tabulce s pravidly z obrázku 4.3 je na obrázku 4.4.



Obrázek 4.4: Postup sl. automatu pro řetězec `B0123E`

Tento automat funguje velice podobně jako klasické dvoudimensionální automaty, používá čtyři směry pohybu a jeho přechody lze popsat jednoduchým algoritmem, který ale není nijak minimalizovaný. Jeho výhodou je vyhodnocení pravidla z obou směrů a možnost použití speciálních zástupných znaků bez ovlivnění fukčnosti automatu. Na druhou stanu automat provádí mnoho nepotřebných přechodů, kterými narůstá výpočetní čas vyhledávání.

4.2.3 Automat jako struktura

Jedním z možných způsobů konstrukce automatu bylo vytvoření struktury, která by tento automat reprezentovala. Nebylo by pak zapotřebí tabulky pravidel a pohyby ve dvou dimenzích by nebyly ve skutečnosti prováděny (pozice v tabulce může být reprezentována stavem automatu). Automat by mohl být reprezentován pouze zápisy pravidel δ funkce, pomocí nichž by byly prováděny přechody mezi stavy.

Forma takového zápisu v textové podobě by byla například ve formátu: {*pozice v obrazu, stav automatu, znak řetězce : následující pozice v obrazu, následující stav automatu* }.

V našem případě by nebylo nutné zapisovat ani pozici v obraze, ta by byla vyjádřena stavem automatu. Ve skutečnosti jde o řádky tabulky přechodů viz sekce 2.3.

Výše zmíněný zápis je mnohem méně objemný, než tabulky pravidel používaná ve výsledné aplikaci, ovšem pro koncového uživatele je zcela nemožné, aby se v takovém zápisu zorientoval. Zápis je neprůhledný a jeho případná dodatečná editace (kvůli měnícím se pravidlům pro RZ pravděpodobná) by přinesla velkou možnost chyb při zápisu, zároveň by byla časově náročná a vyžadovala obsluhu osoby zapojené do vývoje databáze. Editace tabulky pravidel je proveditelná pro koncového uživatele bez znalosti vývoje softwaru, pro usnadnění je vyvinut i editor těchto souborů (viz 6.2).

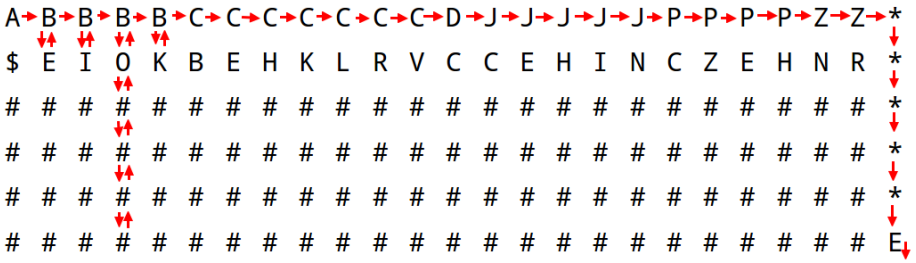
4.2.4 Čtyřsměrný automat

Název automtu připomíná automat $4DFA$, jde ale pouze o podobnost z důvodu použití čtyř směrů vyhledávání v tabulce pravidel. Stavem automatu je jeho pozice v tabulce a na jejích hranicích, finálními stavy je pozice automatu mimo tabulku, konkrátně pod ní. Počáteční stav automatu je pozice $(1, 1)$ v tabulce a pokud se automat nachází na konci vyhledávání vpravo od tabulky, nebylo nalezeno žádné odpovídající pravidlo a řetězec nebyl přijat.

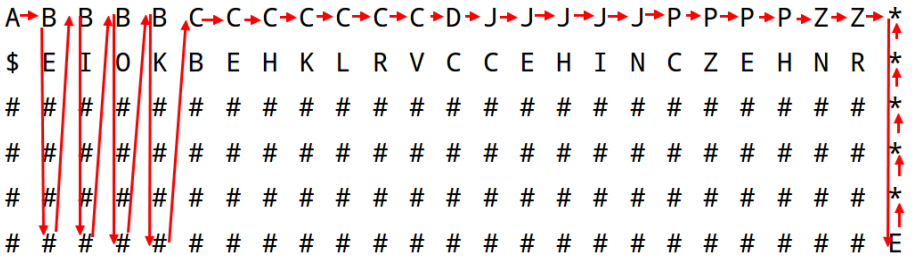
Po spuštění automat prochází prvním řádkem tabulky směrem R a hledá element, který odpovídá prvnímu znaku zadaného řetězce. Pokud je takový element nalezen pokračuje automat směrem D, dokud jeho pozice není mimo tabulku (což značí úspěch), nebo nenachází

element pravidlu neodpovídající, v takovém případě se čtecí hlava automatu navrací na první řádek tabulky a pokračuje opět ve směru R.

Na obrázku 4.5 je značeno úspěšné vyhledávání výše uvedené RZ v tabulce pravidel. Dále demonstrujeme neúspěšné vyhledávání spz 123456 v obrázku 4.6.



Obrázek 4.5: Postup čtyřsm. automatu pro řetězec `BO123E`



Obrázek 4.6: Postup čtyřsm. automatu pro řetězec `123456`

Automat umožňuje pracovat se zástupnými znaky a nikdy nedosáhne neukončujícího stavu viz 2.3, pohyby po tabulce jsou výsledkem jednoduchého algoritmu, který však není příliš efektivní. Jak lze vidět, jsou prováděny neustálé návraty přes odpovídající prefixy pravidel, které ovšem řetězci neodpovídají. Pokud by se automat pohyboval například po tabulce pravidel polských RZ (viz 5.2.3), která vždy obsahuje několik stejných prefixů větší délky, mohlo by toto navracení mít velký vliv na zpomalení chodu programu. Zároveň je pro potřebu návratů na vyšší řádky tabulky nezbytné udržovat celou tabulku v paměti po dobu chodu automatu.

4.2.5 Třísměrný automat

Jak bylo popsáno v předchozí sekci, je často nevýhodné pohybovat se po obrazu ve čtyřech směrech, z nichž jeden je použit jen pro návraty při neúspěšném vyhledávání. Pokud by směr navracení (U) byl odstraněn, nebylo by nadále nutné udržovat v paměti řádky výše položené, než je řádek, na kterém se nachází čtecí hlava automatu. Dále by z hlediska aplikace bylo nutné udržovat v paměti pouze jeden řádek tabulky v jednom čase. Při přechodu automatu směrem D by byl načten další řádek, který by v paměti přepsal řádek předchozí.

Porovnání třísměrných automatů s jinými bylo provedeno např. v práci [5] a skutečnost, že automaty s méně směry jsou schopné efektivnější práce, je zřejmá už ze sekce 4.1.2. Proto je ve výsledné aplikaci nasazen třísměrný automat s pamětí pro podmnožiny indexů.

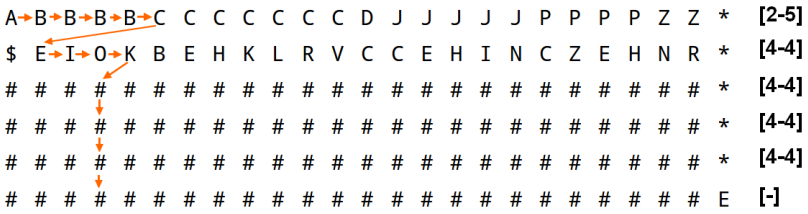
Paměť automatu má jednoduchý význam, automat vždy projde řádek a zapamatuje si sloupce, které jsou kandidáty na pravidlo pokrývající vstupní řetězec.

Automat s pamětí intervalů indexů

Tento automat prochází tabulkou ve třech směrech (L,R,D) a vždy pracuje pouze s jedním řádkem tabulky. Počáteční pozice automatu v tabulce je $(1, 1)$, automat pak ve směru R prochází prvním řádkem tabulky. Při nálezu elementu, odpovídajícího prvnímu znaku řetězce, na pozici $(1, n)$ automat vloží do své vnitří paměti údaj n a pokračuje dále ve směru R.

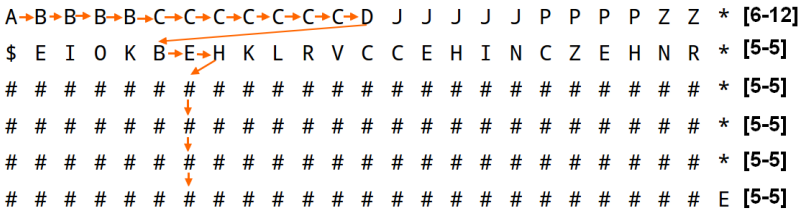
Po nalezení elementu, který jako první neodpovídá znaku řetězce, na pozici $(1, m)$ (přičemž $n < m$) automat ukládá do své paměti údaj $m - 1$ a pokračuje směrem D a odpovívajícím počtem kroků L na pozici $(2, n)$. Takto je analyzován každý řádek tabulky, dokud není paměť automatu prázdná, či automat není schopen uskutečnit krok směrem R (neúspěšné vyhledávání), nebo automat nedosáhne konce tabulky bez možnosti uskutečnit krok D (úspěšné vyhledávání).

Demonstrace činnosti automatu je vidět na obrázcích 4.7, 4.8 a 4.9. Obsah paměti automatu je vždy vyobrazen v pravé části obrázku.



Obrázek 4.7: Postup třísm. automatu pro řetězec `BO123E`

Obrázek 4.8: Postup třísm. automatu pro řetězec `KA1234`



Obrázek 4.9: Postup třísm. automatu pro řetězec `CE1234`

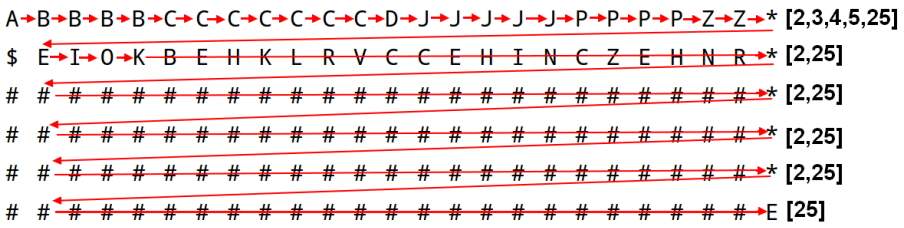
Jak lze vidět na obrázku 4.7, tento automat nemůže pracovat se zástupnými znaky, nelze totiž seřadit pravidla tak, aby zástupné znaky byly vždy na pozici, která bude zahrnuta do intervalu odpovídajících pravidel. Proto je použit další definovaný automat.

Automat s pamětí množin indexů

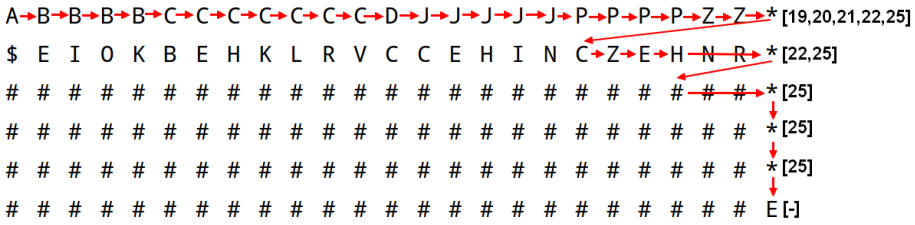
Tento automat je použit ve výsledné aplikaci z důvodů jeho předností jako je odstranění nutnosti udržovat celou tabulku pravidel v paměti, možnost práce se zástupnými znaky a odstranění nadbytečných pohybů po tabulce.

Činnost automatu je podobná činnosti automatu předchozího. Rozdíl je takový, že první řádek tabulky je vždy analyzován (směrem R) celý. Je vytvořena množina indexů, na nichž byly nalezeni kandidáti na pravidla odpovídající řetězci. Na dalších řádcích tabulky se vždy pokračuje s touto množinou, z níž je vytvořena podmnožina indexů, které na daném řádku opět odpovídají řetězci. Koncem vyhledávání je vyprázdnění množiny indexů (neúspěšné vyhledávání), nebo dokončení průchodu tabulkou (úspěšné vyhledávání). Stavem takového automatu je ve skutečnosti mocnost množiny indexů, která je uložena ve vnitřní paměti automatu.

Demonstrace vyhledávání je na obrázcích 4.10 a 4.11.



Obrázek 4.10: Postup třísm. automatu pro řetězec `BO123E`



Obrázek 4.11: Postup třísm. automatu pro řetězec `PHA123`

Tento automat se jeví jako nejlepší volba pro cílové využití v aplikaci. Testování a popis algoritmu automatu viz kapitola 6. V příkladech lze vidět velké skoky po tabulce, které však v programu nejsou prováděny. Z hlediska náročnosti v měřítku výpočetního času se algoritmus ukázal jako nejefektivnější.

Kapitola 5

Registrační značky

V této kapitole se zabýváme teorií registračních značek (dále RZ). Jsou analyzována pravidla značení aut v různých státech Evropské unie, speciální případy značení vozidel a zasazení těchto pravidel do reprezentace využité ve vyvíjené aplikaci pro určení státní příslušnosti zadané RZ. V následující kapitole jsou tyto poznatky použity pro vytvoření aplikace a databáze obsahující všechna pravidla pro RZ daných států.

5.1 RZ v Evropě

Výsledná aplikace je určena pro české uživatele, proto je databáze značek orientovaná převážně na Českou republiku, státy sousedící s ČR a státy Evropské unie, nebo státy nacházející se v Evropě s podstatným podílem na dopravě v České republice, což jsou následující státy:

Nejbližší státy

Česká republika, Slovenská republika, Německo, Polsko, Rakousko

Státy Evropské unie

Belgie, Bulharsko, Estonsko, Francie, Chorvatsko, Itálie, Litva, Lotyšsko, Maďarsko, Nizozemsko, Rumunsko, Řecko, Slovinsko, Velká Británie, Španělsko

Ostatní státy

Bělorusko, Bosna a Hercegovina, Makedonie, Moldavsko, Rusko, Srbsko, Ukrajina

5.1.1 Štítky se zkratkou státu

Převážná většina zde analyzovaných států aktuálně používá RZ, které obsahují tzv. *euroštítek*, určující jejich státní příslušnost. Z toho důvodu se vývoj aplikace, která určí státní příslušnost vozidla z textu RZ, zdá jako zbytečný. Ve skutečnosti je ale velké procento RZ vydáno před zahájením tohoto značení státní příslušnosti a zároveň je nutné zvážit způsob, jakým jsou RZ zachyceny například na záznamech kamer. Ty jsou často kvůli ostrému obrazu textu RZ černobílé, což podstatně snižuje čitelnost štítku. Dalším faktorem je nižší kvalita těchto snímků, proto jsou často štítky zcela nečitelné. Ukázka takového záznamu RZ je na obrázku 5.1.

Obrázek 5.1: Fotografie RZ s nízkou kvalitou (zdroj: http://www.olavsplates.com/)

5.2 Pravidla ve státech

V této sekci popisujeme pravidla pro tvary registračních vozidel v České republice a sousedních státech. Jsou analyzovány i velmi raritní tvary. Především je ovšem kladen důraz na tvary pro nákladní automobily z důvodu jejich častého výskytu na českých silnicích.

Aplikace se nezabývá tvary RZ pro motocykly, traktory, či jinou techniku, jenž nepodléhá pravidlům pro standardní motorová vozidla a tvary RZ na zakázku, které často nepodléhají žádným pravidlům.

5.2.1 Česká republika

Na tento stát, jakožto domovský pro aplikaci, je kladen nejvyšší důraz z hlediska úplnosti databáze, proto se tato sekce zabývá i stručnou historií RZ v ČR, stále platnými staršími tvary RZ a speciálními tvary RZ. Jako zdroj byla použita vyhláška Ministerstva dopravy ČR [3] a kniha [11].

Starší tvar RZ stále se vyskytující na českých silnicích je veden od roku 1960. Tvar RZ byl složen z dvoupísmenného kódu okresu a čtyř číslic. Tento tvar byl nedostačující počtem kombinací, proto od roku 1967 bylo zavedeno třetí sériové písmeno. V aplikaci je pro tyto tvary zavedeno 98 pravidel pro jednotlivé okresy. Speciálním pravidlem se vyznačuje okres hlavního města Prahy, kde je okres označen pouze prvním písmenem A (neexistuje jiný okres s tímto prefixem), poté následují dvě sériová písmena a čtyři číslice.

Výjimečná pravidla pro tyto kombinace jsou zapříčiněna cenzurou minulého režimu a jejich souvislostí v českém jazyce. Pro okres Přerov (zkratka PR) byla vynechána sériová písmena C a D a okres Strakonice (zkratka ST) bylo vynecháno sériové písmeno B.

Nynější tvar RZ je určen vyhláškou Ministerstva dopravy `č. 243/2001 Sb.` uvedenou v platnost roku 2001 [3]. Oficiálně jsou tyto tvary označovány jako tvary 101,103,105,106,115 a 116. Tvar je inspirován předchozím, neobsahuje však zkratky okresů z důvodu zrušení okresů jako oficiálního územního celku. Původní vyhláška stanovuje tvar jako sériovou číslici následovanou písmenem kraje a další sériovou číslicí, poté následuje posloupnost čtyř sériových číslic. Písmena odpovídající krajům jsou následující: **A**, **B**, **C**, **E**, **H**, **J**, **K**, **L**, **M**, **P**, **S**, **T**, **U** a **Z**.

Kvůli nedostatečnosti tohoto schématu je nyní vyhláška upravena a kromě sériového čísla vpravo od písmene kraje je možné používat sériové písmeno stejně jako v předchozím tvaru z roku 1967. Tento tvar je uveden v platnost od roku 2009 v Praze a následovně ve Středočeském a Jihomoravském kraji. Aplikace tyto změny reflektuje a pro zabránění nutnosti úpravy databáze v následujících letech je tento tvar podporován pro všechny kraje. V aplikaci je takto vytvořeno 14 pravidel, kde je tvar RZ definován jako číslo, písmeno kraje, libovolný znak a další čtyři čísla. Procesem vydávání RZ se zabývá práce [13], kde je poukázáno na skutečnost, že sériové číslice nejsou vydávány podle posloupnosti žádostí

o registraci vozidla, jak tomu bylo v předchozím značení.

Následuje výčet **speciálních tvarů** českých RZ:

trvale manipulační vozidla tisknuto zelenou barvou, pouze písmeno kraje a 4 číslice,
vozidla pro export 5 libovolných znaků, písmeno E a červený štítek platnosti,
vozidla pro osobu s imunitou tisknuto modře, 3 číslice, písmena `CD` a dvě číslice,
vozidla pro diplomatické osoby tisknuto modře, 3 číslice, písmena `XX` a dvě číslice,
vozidla pro diplomatické mise tisknuto modře, 3 číslice, písmena `XS` a dvě číslice,
vozidla pro honorární konzuly tisknuto modře, 3 číslice, písmena `HC` a dvě číslice,
vozidla ve zkušebním provozu písmeno `F` následované čtyřmi číslicemi,
vozidla historická dvě číslice, písmeno `V` a čtyři číslice,
vozidla sportovní dvě číslice, písmeno `R` a čtyři číslice

Přípustné délky	Počet pravidel	Paměť pro data
5, 6, 7	232	7,8 kB

Tabulka 5.1: Statistika Česká republika

5.2.2 Slovenská republika

Slovenská historie RZ je spjata s historií České republiky. Nový systém značení slovenských RZ ale nepodporuje bývalé tvary minulého režimu, proto všechny tvary dnes používané slovenskými vozidly jsou vyhrazeny vyhláškou slovenského Ministerstva vnitra z roku 1997, která byla upravována pouze z hlediska grafické podoby RZ. Zdrojem informací byl vzorník slovenského Ministerstva vnitra [2].

Současný **standardní tvar** RZ pro vozidla je rozdělen podle dvou písmen reprezentujících okres vydání příslušné RZ, kdy jeden okres může být podle velikosti reprezentován více zkratkami (např. Bratislava: `BA, BD, BE, BI, BL, BT`). Po této zkratce následuje posloupnost tří číslic a dvou písmen. Proces číslování vydávaných RZ je následující: zkratka kraje následovaná nejprve kombinací `000AA`, další tvar ve výrobě je pak `001AA`, pro vypršení číselných kombinací je navýšeno sériové písmeno tedy po `999AA` následuje `000AB` a pro `999AZ` následuje `000BA`. Pro větší okresy po využití všech kombinací v sérii je použita další zkratka pro okres.

Omezení pro tento tvar RZ jsou malá, omezuje je pouze 79 okresů státu a mírně vyšší počet kombinací zkratek okresů. Počet těchto zkratek bude zřejmě růst, proto budou v budoucnosti nutné úpravy databáze. V současné době nelze přepokládat, jaké tvary zkratek budou dále povoleny. Nyní databáze obsahuje 91 pravidel pro tyto RZ se zástupnými znaky pro sériové číslice a písmena.

Speciální tvary slovenských RZ obsahují RZ přidělené *vozidlu určenému na jednotlivý vývoz* s tvarem začínajícím zkratkou okresu, následovanou písmenem V a třemi číslicemi. Podobný tvar nabývají i RZ přidělené *zkušebním vozidlům*, pouze s třetím písmenem `M`, dále RZ přidělené *sportovním a historickým vozidlům* s písmenem `S`, nebo `H`. Pro *vozidla ministerstev*, či policie je přiřazena RZ skládající se z písmene `P` následovaným pěti číslicemi, podobný tvar s písmenem `C` má značka pro *dovezená vozidla*. Slovenská *vojenská auta* nesou označení obsahující pouze 7 číslic bez jakýchkoliv písmen. *Diplomatická auta* jsou na RZ označena písmeny `EE` nebo `ZZ` následovanými pěti číslicemi.

Přípustné délky	Počet pravidel	Paměť pro data
6, 7	208	10,2 kB

Tabulka 5.2: Statistika Slovensko

5.2.3 Polsko

Polsko jako stát s 16 vojvodstvími a zhruba 300 okresy má jeden z nejrozsáhlejších podílů na databázi aplikace. Pro každý z okresů existuje několik druhů RZ s odpovídající zkratkou okresu. Současný systém RZ značení je platný od roku 1976 s grafickými obměnami. Jako zdroj dat uvádíme vyhlášku polského ministerstva dopravy o značení a registraci vozidel [1].

Kvůli obsáhlému výpisu pravidel **standardních tvarů** RZ zde zavádíme popis pomocí jednoduché gramatiky, kde `<V>` značí znak zkratky vojvodství, `<OK>` značí znak zkratky okresu, `<num>` značí libovolné číslo, `<num1>` značí libovolné číslo kromě nuly a `<char>` libovolné povolené písmeno. V tomto schématu se tedy RZ skládá z provinciálního označení a posloupnosti sériových číslic a písmen.

- `<V><OK><num><num><num><num><num>`
- `<V><OK><num><char><num><num><num>`
- `<V><OK><num><char><char><num><num>`
- `<V><OK><num><num><num><num><char>`
- `<V><OK><num><num><num><char><char>`
- `<V><OK><OK><char><num><num><num>`
- `<V><OK><OK><num><num><char><char>`
- `<V><OK><OK><num1><char><num><num>`
- `<V><OK><OK><num><num><char><num1>`
- `<V><OK><OK><num1><char><char><num1>`
- `<V><OK><OK><char><char><num><num>`
- `<V><OK><OK><num><num><num><num><num>`
- `<V><OK><OK><num><num><num><num><char>`
- `<V><OK><OK><num><num><num><char><char>`

Tato gramatika je v obměně použita v databázi se všemi přípustnými kombinacemi a minimalizacemi (např. poslední dvě pravidla lze spojit v jedno). Omezení se vyskytuje v případech číslic, které nemohou být 0 a povolených písmen, tedy libovolnými písmeny abecedy vyjma `B, D, I, O, Q` a `Z`. Pravidel vztahujících se k těmto tvarům jsou řádově stovky.

Mezi **speciální tvary** v Polsku patří RZ pro *historické vozy* s dvěma až třemi písmeny pro značení okresu, jednou nebo dvěma číslicemi a jedním písmenem. *Vozidla pro export* nesou RZ s písmenem vojvodství následovaným pěti číslicemi, či čtyřmi číslicemi a jedním písmenem na konci. Výjimku tvoří písmeno `B`, pokud RZ tohoto tvaru končí tímto písmenem, jde o *vozidlo testovací*. *Diplomatické značení* zahrnuje písmeno vojvodství a šest následujících číslic. RZ začínající na skupiny písmen HA, HB, HC, HK, HM, HP,HS, HW, nebo U s navazující posloupností čtyř až pěti znaků značí *vozidla veřejné správy*.

Přípustné délky	Počet pravidel	Paměť pro data
5, 6, 7, 8	1983	15,2 kB

Tabulka 5.3: Statistika Polsko

5.2.4 Německo

Německo stejně jako Polsko má na svém území velký počet okresů a obdobně obsáhlý je výčet pravidel pro **standardní** RZ vozidel. Oproti Polsku není pravidel pro jeden okres neúnosné množství pro slovní popis, tvary jsou tedy následující. Označení okresu se skládá z jednoho až tří písmen, následuje jedno až dvě písmena sériová a jedna až čtyři číslice. Problém zde nastává s německými speciálními znaky, tedy znaky přehláskovanými (ö, ü, ...). V databázi jsou zaměněny za jejich nepřehláskované ekvivalenty.

Standardní německá RZ nabývá délky od 4 do 8 znaků. Celkově je ve všech délkách těchto pravidel 3002. Jako zdroj pro vytváření pravidel sloužila databáze na stránce http://www.kba.de/DE/Fahrzeugtechnik/Zum_Herunterladen/zum_Herunterladen_node.html.

Speciální vozidla v Německu jsou často odlišena pouze pomocí nálepek na RZ, proto pro ně nezavádíme dodatečná pravidla, v praxi jsou německá auta odlišována hlavně speciálním písmem, jímž je ražena RZ. V databázi jsou tedy speciální tvary pokryty pravidly pro standardní RZ.

Přípustné délky	Počet pravidel	Paměť pro data
4, 5, 6, 7, 8	3002	21,7 kB

Tabulka 5.4: Statistika Německo

5.2.5 Rakousko

Rakouské **standardní RZ** se velmi podobají slovenským, obsahují jeden nebo dva znaky pro značení regionu, kterých je přibližně sto. Následuje posloupnost číslic a písmen. Bohužel není zde přítomno žádné striktní pravidlo pro tvar sériové části RZ, platí pouze pravidlo o výskytu minimálně jednoho písmena a jedné číslice, z toho důvodu obsahuje záznam pro Rakousko 2294 pravidel, které však pokrývají i vozidla **speciální** pro veřejné služby a výjimečně i RZ na zakázku.

Je pokryta jakákoliv kombinace sériových znaků, která splňuje výše zmíněná pravidla a spolu s náležící zkratkou okresu splňuje pravidlo délky RZ, což je délka od 3 do 7 znaků včetně znaků pro okres. Zdrojem pro pravidla byla databáze dostupná z https://www.kfz.net/autorecht/autokennzeichen/oesterreich/. Pro tento typ RZ se dvoudimensionální výčet pravidel nejeví jako nejefektivnější řešení, v aplikaci ovšem zamýšlíme jednotný přístup ke všem státům, proto nebylo použito žádných postranních mechanismů pro minimalizaci.

Přípustné délky	Počet pravidel	Paměť pro data
3, 4, 5, 6, 7	2294	15,8 kB

Tabulka 5.5: Statistika Rakousko

V této kapitole byla představena pravidla pro RZ států sousedících s Českou republikou. Ostatní státy zmíněné v sekci 5.1 jsou v databázi analyzovány také, ovšem zde by výčet jejich pravidel byl příliš objemný. Některé tyto státy a obrázky pravidel analyzovaných v této kapitole jsou znázorněny v příloze B. Kompletní obsah databáze zahrnuje přibližně 9 000 pravidel pro 27 států. Všechna pravidla pro RZ jsou v databázi zastoupena pomocí tabulek pravidel (viz 6.1.2). Mimo jiné jsou v databázi řešeny i problémy odlišných abeced, jako jsou azbuka a řecká abeceda.

Kapitola 6

Aplikace

Tato kapitola představuje spojení předchozí teorie v aplikaci vyvíjené v rámci této práce. Kromě aplikace pro vyhledávání států s pravidly pro RZ, jimž odpovídá vstup byla dodatečně vyvinuta aplikace pro editaci vytvořené databáze. Kapitola také popisuje databázi, která byla vytvořena z tvarů RZ všech států uvedených v kapitole 5.

6.1 Aplikace pro vyhledávání

Tato aplikace je zamýšlena pro použití v rámci klasifikace RZ v mezinárodním měřítku. Především tedy při auditu mýtného na dálnicích, kde může být použita jako podpůrný nástroj pro automatickou klasifikaci společně s nástroji umožňujícími extrahovat text RZ ze snímků kamer (viz 6.1.1).

Aplikace by také mohla najít uplatnění v rámci bezpečnostních systémů, jako jsou veřejné kamerové systémy, nebo kamery na parkovištích, kde by umožňovala bezpečnostním složkám identifikaci národnosti řidiče daného vozidla. V tomto ohledu by aplikace mohla být použita i při pronásledování vozidla.

Aplikace ale může být použita i pro zájmové účely, jako je statistika zabývající se výskytem řidičů různých národností na silnicích, nebo pouhý zájem uživatele o to, jaké národnosti je řidič sousedního automobilu, či automobilu, jenž uživatelovi zapříčinil škodu.

6.1.1 Existující řešení

Autoři si nejsou vědomi žádné aplikace, která umožňuje dosáhnout zjištění státní příslušnosti automobilu pouze pomocí zadání znaků RZ. Pokud takové aplikace existují, jsou určeny pro interní použití. Proto jsou představeny aplikace, které by umožňovaly společné použití s vyvinutou aplikací. Tedy aplikace extrahující text RZ z video záznamu a aplikace umožňující klasifikaci RZ ve vnitrostátním měřítku, tedy detekci kraje, ve kterém je automobil registrován.

Aplikace pro rozeznávání textu

Tyto aplikace slouží pro detekci RZ na záznamu a extrakci znaků v obrazu do textové podoby. Obecně je použito technologie OCR (Optical Character Recognition), která umožňuje klasifikaci jednotlivých obrazců jako textu (viz [6]).

Na Vysokém učení technickém v Brně bylo vyvinuto několik aplikací v rámci závěrečných prací, které tento účel splňují explicitně pro rozeznávání textu RZ.

Aplikace vyvíjená v rámci práce [15] umožňuje detekovat přítomnost RZ v obraze a pracuje s úpravami obrazu, jenž umožňují eliminovat rušivé elementy jako je špatná kvalita snímku, či rozmazání. Nicméně tato práce není připravena na různé tvary RZ, jako jsou nestandardní velikosti RZ, či například černý podklad RZ, používaný v Rakousku a dříve v Polsku.

Aplikace vyvinutá v rámci práce [10] umožňuje zároveň detekci přítomnosti RZ obraze a extrakci textu RZ. Výsledná aplikace se velmi dobře vyrovnává s podobností znaků, například "7" a "Z". V práci je však uvažováno pouze standardních tvarů českých RZ (viz 5.2.1). Proto by aplikace byla nepoužitelná pro spojení s vyvíjenou aplikací, výskyt jiného počtu než sedmi znaků by vedl na nedefinované chování aplikace.

Existuje více prací zabývajících se tímto tématem, zmínění zástupci však reprezentují největší nedostatky, což je nepřipravenost programů na špatný stav RZ, jako je sníh, či špína na RZ, vysoká rychlost vozidla, špatný úhel snímku a použití aplikace pouze v rámci českých RZ.

Jako zástupce volně šiřitelných aplikací představujeme aplikaci *License Plate Reader (ANPR)* (dostupné na: `https://play.google.com/store/apps/details?id=com.itzkow.licenceplatereader`), která pomocí fotoaparátu mobilního telefonu umožňuje extrahování textu ze zachycené RZ.

Aplikace byla testována na několika tvarech RZ více států a bylo zjištěno, že aplikace podporuje libovolné tvary RZ. Opět ale rozeznávání textu není dokonalé například na obrázku 6.2 lze vidět špatně opsaný text a na obrázku 6.2, kde je snímána RZ ruského diplomata, tedy bílá písmena na červeném pozadí (viz sekce B.10), aplikace není schopna extrahovat text.

Obrázek 6.1: Špatně opsaný text

Obrázek 6.2: Špatná extrakce textu

Aplikace pro klasifikaci RZ

Následující aplikace jsou volně dostupné aplikace z obchodu *Google Play*. Umožňují převážně pouhé určení oblasti státu, v níž je automobil registrován. Ve spojení s vyvíjenou aplikací by v daných státech mohlo být umožněno získání těchto bližších údajů o původu automobilu.

Aplikace *Number Plates* (dostupná z: https://play.google.com/store/apps/details?id=com.regioneu) umožňuje získání informací o zkratkách regionů v těchto státech: Rakousko, Bulharsko, Česká republika, Německo, Francie, Velká Británie, Itálie, Polsko, Rumunsko, Slovensko, Slovinsko. Na obrázku 6.3 lze vidět, že menu aplikace obsahuje i další státy, proto se domníváme, že současný obsah databáze není konečný. Po výběru státu se objeví seznam zkratek regionů, jak je vidět na obrázku 6.4.

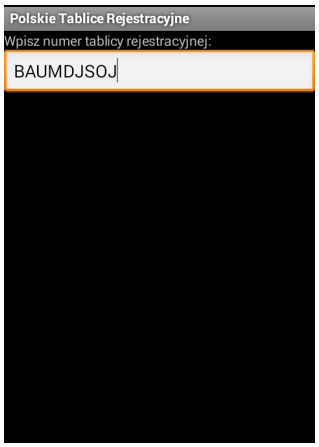
Obrázek 6.3: Menu aplikace

Obrázek 6.4: Zkratky pro ČR

Databázi těchto tvarů hodnotíme jako obsáhlou, ovšem jak je vidět z obrázku 6.3, je nutné vybrat příslušný stát před zadáním tvaru RZ. Aplikace by mohla pro příslušné státy obsahovat pravidla jako vyvíjená aplikace, pro automatické určení státní příslušnosti, bylo by možné s autorem aplikace tuto možnost konzultovat.

Aplikace *Polskie Tablice Rejestracyjne* (dostupná z: https://play.google.com/store/apps/details?id=pl.araneo.ptr) je orientována pouze na použití spojené s polskými RZ, což zahrnuje cca 300 různých tvarů (viz 5.2.3). Aplikace ovšem umožňuje kontrolu, zda je zadaný řetězec polskou RZ, na obr 6.5 je zadán plný tvar polské RZ a na obr 6.6 je zadán řetězec, který pravidla pro polské RZ nesplňuje. Aplikace tedy pracuje s pravidly pro tvary RZ, ale pouze v rámci jednoho státu.

Obrázek 6.5: Platný tvar RZ

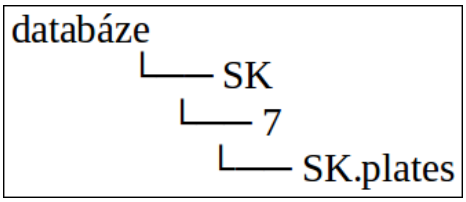


Obrázek 6.6: Neplatný tvar RZ

6.1.2 Databáze aplikace

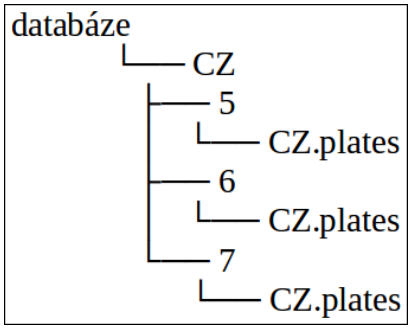
V sekci byla znázorněna podoba tabulky pravidel, ve které aplikace vyhledává odpovídající pravidlo, zde je znázorněno umístění těchto tabulek v databázi souborů.

Každá tabulka je reprezentována souborem, na jehož umístění v databázi má vliv stát, jehož pravidla soubor obsahuje, a délka těchto pravidel. Na obrázku 6.7 je znázorněno umístění souboru `SK.plates`, jenž obsahuje pravidla o délce 7 znaků pro stát Slovensko.



Obrázek 6.7: Umístění `SK.plates`

Jednotlivé státy v databázi jsou reprezentovány složkami, jejichž názvy odpovídají zkratkám států, tyto zkratky jsou uloženy v JSON konfiguračním souboru `statesdict.json`, který je načten vždy při startu aplikace. Obsahem složek států jsou složky, jejichž názvy odpovídají délkám pravidel obsažených tabulek, příklad pro Českou republiku na obrázku 6.8.



Obrázek 6.8: Složka `CZ` v databázi

Obsah složky pro délku pravidla je následující:

- soubor (soubory) s tabulkou pravidel (`plates` soubory)
- složka `orig` s originálními soubory obsahujícími nezpracovaná pravidla
- složka `imgs` se spojovacími soubory (`imgs` soubory) pro spojení pravidel s obrázky

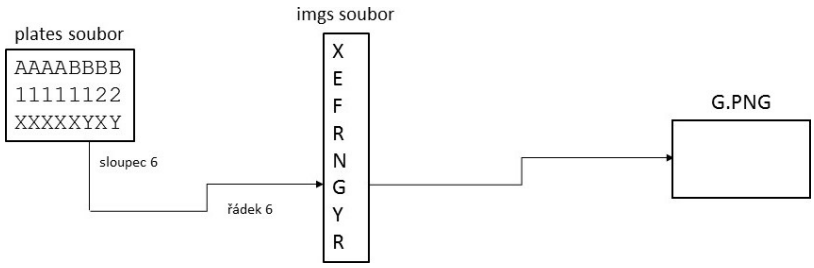
Soubor obsahující tabulku pravidel má tvar zobrazený na obr 6.9. Obsahuje povinně řádek s počtem pravidel v souboru pro kontrolu konzistence databáze, případně řádek s definicí speciálních znaků pro daná pravidla (viz 6.1.3). Poté je tvar souboru stejný jako tabulky zobrazené na obrázku 4.3, obsahuje sloupce reprezentující jednotlivá pravidla.

```
{'*': 'ACEFGHJKLMNPRSTUVXYZ123456789', '@' :
'123456789'}
49
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB$
IILLSSAAABBBGGGHHHIIIKKKLLLMMMSSSSSSSSSSSSWWWZZZ#
######UUUIIIRRRAAAAAALLLMMMNNNEEEIIIKKKUUUMMMAAA#
#*#*#**#@*#@*#@*#@*#@*#@*#@*#@*#@*#@*#@*#@*#@*#@#
#*#*#**#**#**#**#**#**#**#**#**#**#**#**#**#**#**#*#
*#*#*##**#**#**#**#**#**#**#**#**#**#**#**#**#**#**#
*#*#*##*@#*@#*@#*@#*@#*@#*@#*@#*@#*@#*@#*@#*@#*@#
```

Obrázek 6.9: Soubor `PL(B)` s délkou pravidel 7

Ve složce `orig` jsou soubory, které obsahují nezpracovaná pravidla. Slouží pro editaci a vytvoření příslušných `plates` souborů. Jejich editace je mnohem jednodušší, pravidla jsou reprezentována řádky a nejsou obsaženy žádné dodatečné informace. Vytvoření `plates` souboru pak vykonává editor databáze (viz 6.2).

Ve složce `imgs` jsou soubory s příponou `imgs`, které slouží k nalezení odpovídajících PNG souborů pro zobrazení v GUI aplikace. Každý řádek souboru představuje jméno souboru, který obsahuje obrázek pravidla s odpovídajícím indexem v souboru `plates`. Propojení je znázorněno v obrázku 6.10.



Obrázek 6.10: Propojení souborů

Názvy souborů obsahují vždy zkratku příslušného státu a v případě obsáhlých tabulek pravidel (Německo, Polsko, Rakousko) případně i první písmeno pravidel, z toho důvodu, že jediný soubor pro všechny pravidla určité délky obsahoval stovky až tisíce sloupců. Této výjimky je tedy použito pro minimalizaci doby hledání pravidla a úsporu paměti.

Cesty k příslušným souborům jsou tedy následující:

- soubor s pravidly délky 5 ve státě Česká republika:
  `database/CZ/5/CZ.plates`
- soubor s pravidly délky 8 ve státě Polsko s prvním písmenem K:
  `database/PL/8/PL(K).plates`

6.1.3 Program

Výsledná aplikace je vytvořena v jazyce Python a má podobu GUI umožňujícího zadání vyhledávané RZ a zobrazení výsledků. Vstup je možné zadat v jakémkoli tvaru, který obsahuje písmena a čísla povolené abecedy a libovolné množství bílých znaků. Při drobné úpravě zdrojového kódu může být aplikace použita i v příkazové řádce, umožňující dávkové zpracování vstupu.

Obrázek 6.11: Aplikace po startu

Po spuštění aplikace je možné okamžitě zadávat vstup bez nutné interakce s myší. Spuštění vyhledávání je možné pomocí stisku klávesy `Enter`, nebo stisknutím tlačítka *Vyhledat* v GUI aplikace. Výsledek je vystaven v oboru milisekund.

Obrázek 6.12: Výsledek vyhledávání `HOA 123`

Jak lze vidět na obrázku 6.12, je možné zobrazit až 5 výsledků hledání, toto číslo je dostatečné, jelikož pomocí experimentů bylo zjištěno, že vyššího počtu výsledků se díky rozdílnosti pravidel jednotlivých států nedosahuje. Jednotlivé výsledky obsahují vlajku příslušného státu, název státu, podobu splněného pravidla a obrázek korespondující s pravidlem.

Vyhledávání výsledků zahrnuje tyto fáze:

1. Úprava vstupu (odebrání bílých znaků) a vyhodnocení délky vstupu
2. Vytvoření seznamu států, které obsahují pravidla se stejnou délkou, jako je vstup
3. Prohledávání tabulek pravidel jednotlivých států (funkce automatu)
4. Extrakce případných odpovídajících pravidel
5. Nalezení souborů s obrázky pravidel
6. Vytvoření slovníků obsahujících položky výsledků
7. Zobrazení výsledků v rámci GUI

Úprava vstupu je banální operace, proto nepotřebuje vysvětlení, probíhá však i kontrola jednotlivých znaků vstupu a případné oznámení neplatného vstupu při použití nepovolených znaků (viz 6.1.5).

Seznam států

Seznam států je reprezentován dvouúrovňovým seznamem `lens`, který je uložen v souboru `lens.json`. Jak značí přípona souboru, data jsou uložena ve formátu JSON. Je přepokládáno, že tyto data jsou výsledkem činnosti editoru databáze. Prvky tohoto seznamu jsou samy seznamem, který obsahuje zkratky všech států s pravidly příslušné délky. Tedy první prvek seznamu obsahuje zkratky států, které podporují RZ s délkou 1, atd.

Vyhledávání

Po získání seznamu států je pro každý stát v tomto seznamu provedeno nalezení příslušného souboru s pravidly (viz 6.1.2). Soubor je čten po řádcích pro úsporu použité paměti.

Prvním řádkem souboru může být zápis slovníku, jenž obsahuje prvky představující speciální znaky a znaky abecedy. Význam slovníku je takový, že přítomný speciální znak

je znakem zástupným pro konkrétní znaky abecedy (viz obrázek 6.9). Speciální znaky používané standardně ve všech státech byly uvedeny v sekci 4.2, tedy ⋆ značí libovolný znak, $ značí libovolné písmeno a ♯ značí libovolnou číslici. Tvar zápisu takového slovníku je následující pro znak ♯: {♯:0123456789}.

Každý soubor s tabulkou pravidel obsahuje řádek s číslicí představující počet obsažených pravidel, toto číslo je generováno editorem databáze při vytváření souboru a slouží ke kontrole konzistence databáze. Případné nadbytečné sloupce tabulky pravidel, doplněné jinak než editorem, jsou ignorovány.

Při načtení prvních řádků souboru pak probíhá samotné vyhledávání v tabulce, které reprezentuje použití automatu. Algoritmus vyhledávání odpovídajícího sloupce tabulky podle automatu v sekci 4.2.5 je následující:

Algorithm 1 Hledání sloupce v tabulce pravidel

Require: soubor `S`, řetězec `str`, slovník speciálních znaků `dic`
row = první řádek tabulky v souboru S
a= a_1 = prázdná množina
for písmeno i v row **do**
\\ pohyb ve směru R po prvním řádku tabulky
if i == s[0] nebo s[0] ∈ dic[i] **then**
\\ byl nalezen sloupec, jehož první znak odpovídá řetězci
přidej index i do množiny a
end if
end for
idx=0
r=další řádek souboru S
while r není prázdný a idx není index mimo str **do**
\\ cyklus po řádcích souboru S a písmenech řetězce str
\\ načítání dalších řádků představuje pohyb automatu ve směru D
idx++
for index j v a **do**
\\ pohyb ve směru R po aktuálně odpovídajících indexech tabulky pravidel
\\ indexy tabulky, které v předešlých iteracích
\\ neodpovídaly řetězci str jsou přeskočeny
if r[j] == s[idx] nebo s[idx] ∈ dic[r[j]] **then**
\\ příslušný index opět vyhovuje
přidej index j do množiny a_1
end if
end for
$a = a_1$
\\ aktualizace údajů o odpovídajících indexech
if a je prázdná **then**
\\ aktuálně neexistuje žádné odpovídající pravidlo v tabulce
\\ neúspěšné vyhledávání **return** `False`
end if
i++
\\ pohyb ve směru L na první odpovídající index
r=další řádek souboru S
\\ pohyb v tabulce ve směru D
end while
if a není prázdná **then**
\\ nalezeno pravidlo (sloupec tabulky), jenž odpovídá řetězci str
return `True`
end if

Algoritmus 1 je spuštěn pro každý stát, tedy odpovídající soubor ve složce státu. Při

nálezu odpovídajícího pravidla je získán tvar tohoto pravidla a zkratka daného státu.

Vytvoření výsledků

Výsledky pro zobrazení v GUI aplikace jsou přijímány ve tvaru slovníku, tyto slovníky generuje funkce `create_dic` a výsledný slovník má následující složky:

shortcut zkratka státu
state název státu
rule odpovídající pravidlo
flag cesta k souboru obrázku vlajky státu
pic cesta k souboru obrázku pravidla

Zkratka státu a pravidlo jsou získány už během vyhledávání, jiné složky slovníku je ovšem nutné získat.

Název státu je získám pomocí slovníku, jenž obsahuje páry zkratek a názvů souborů, tento slovník je načten při startu programu ze souboru `statesdict.json`, kde byl uložen editorem databáze ve formátu JSON.

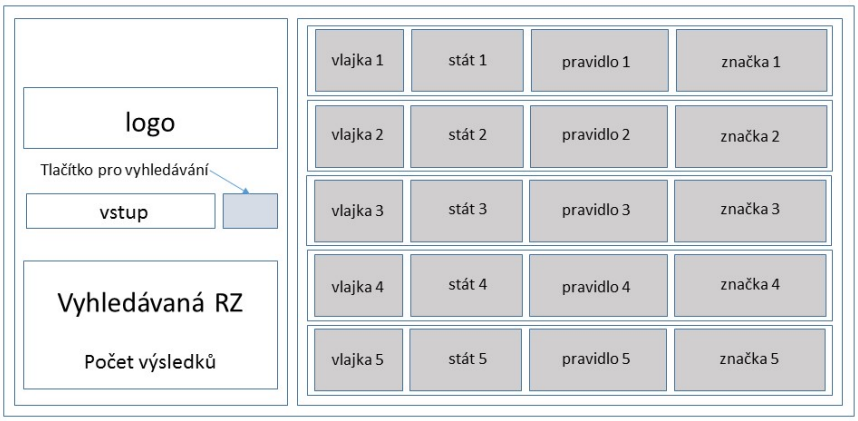
Složka obsahující obrázkové soubory představující vlajky jednotlivých států má vždy odpovídající relativní adresu a jména obsažených souborů odpovídají zkratkám států, proto je tato cesta generována automaticky.

Složka souborů obsahující obrázky pravidel pro jednotlivé státy má také vždy odpovídající relativní adresu a jméno konkrétního souboru je získáno pomocí funkce `find_img` z obsahu souboru `název_státu.imgs` (viz 6.1.2). V tomto souboru je nalezen řádek s indexem odpovídajícím indexu sloupce výsledného pravidla. Tento řádek obsahuje jméno souboru obrázku pro dané pravidlo.

6.1.4 Grafické uživatelské rozhraní

Rozhraní je implementováno pomocí knihovny *Tkinter* (wiki stránky knihovny viz: http://tkinter.unpythonic.net/wiki/). Tato knihovna je objektově orientovaná vrstva implementovaná nad GUI *Tk* a patří k nejrozšířenějším knihovnám pro tvorbu GUI v jazyce Python (často označována jako standardní GUI knihovna pro Python 2 a Python 3). Díky tomu, že knihovna je nadstavbou multiplatformního rámce `Tk`, je možné spustit vytvořené GUI na operačních systémech *UNIX* i *Windows*. Návrh GUI s rozmístěním jednotlivých prvků je na obrázku 6.13.

Jednotlivé prvky prostředí na obrázku 6.13 jsou reprezentovány objekty tříd knihovny *Tkinter*. Rozložení prvků je určeno pomocí jejich ukotvení na strany rodičovských objektů. Nejvyšší rodičovský objekt je reprezentován samotným oknem aplikace. Kvůli nežádoucím deformacím při změnách velikosti okna je tato velikost fixní a to 700×500 pixelů, což je dostatečné pro čitelnost všech informací a zároveň ne příliš velké pro použití na standardních monitorech.



Obrázek 6.13: Návrh GUI aplikace

Okno je rozdělené na dvě části (oblast zadávání a oblast výsledků) pomocí objektů třídy `Frame`, které jsou ukotvené na příslušné strany okna.

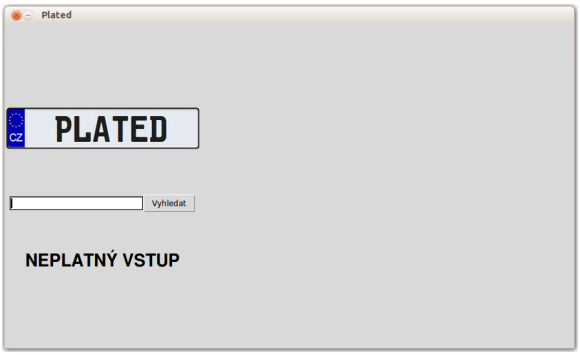
Pravý objekt obsahuje logo aplikace (zobrazeno například na obrázku 6.11), ve středu objektu je poté objekt `in1` (třída `Entry`) sloužící pro zadávání vstupu. Při stisknutí klávesy *Enter* je zadaný text z okna vymazán a je spuštěno vyhledávání, v aplikaci je tato činnost zajištěna funkcí `search`. Stejné chování je spuštěno při zmáčknutí tlačítka *Vyhledat*. Posledním obsaženým objektem je objekt, který obsahuje text shrnující výsledky předchozího vyhledávání.

Levý objekt obsahuje pole objektů třídy `Frame`, přičemž každý prvek tohoto pole reprezentuje jednu položku výsledků. Objekty obsažené v těchto prvcích zobrazují složky výsledků, tedy obrázek vlajky státu, název státu, nalezené pravidlo a obrázek RZ. Pokud je nalezeno méně výsledků než pět, jsou nadbytečné objekty vyjmuty z prostředí, tuto funkčnost zajišťují funkce `spam_w` a `search`.

6.1.5 Příklady použití a testování

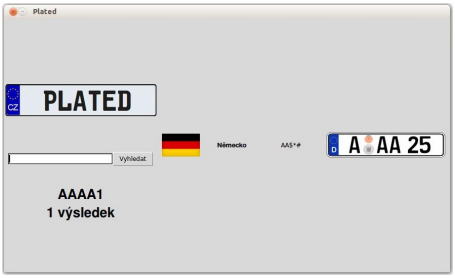
V tomto oddílu představujeme standardní situace a chyby při používání aplikace. Mimo tyto případy byla aplikace testována koncovými uživateli se zaměřením na správnost výsledků a zároveň byla dávkově zpracována celá databáze pravidel pro ověření její konzistence a dotupnosti všech obsažených pravidel.

Je nutné aby aplikace reagovala korektně i na vstupy jako je prázdný vstup, nebo příliš dlouhý vstup, v takovém případě je korektním chováním nahlášení této skutečnosti uživateli skze GUI, jak je vidět na obrázku 6.14. Aplikace také nepodporuje vstupy se znaky speciálních abeced, jako je diakritika, pomlčky, azbuka a přehláskované znaky.

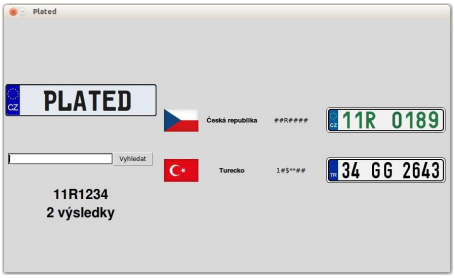


Obrázek 6.14: Chybný vstup

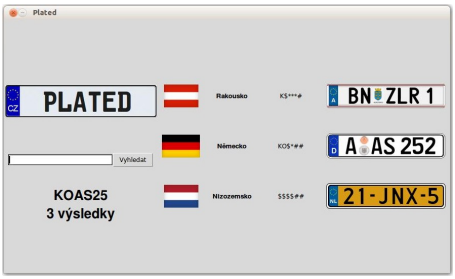
Z oblasti testování grafického uživatelského rozhraní uvádíme korektní vzhled aplikace při jakémkoli výsledku na obrázcích 6.15, 6.16, 6.17 a 6.18. Text oznamující počet výsledků je skloňován podle počtu výsledků.



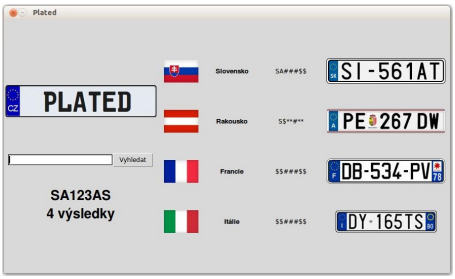
Obrázek 6.15: Vyhledávání `AAAA1`



Obrázek 6.16: Vyhledávání `11R1234`

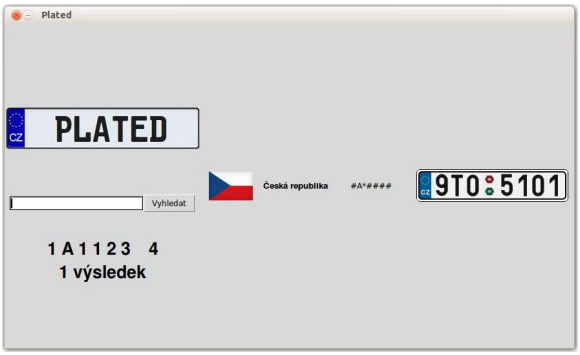


Obrázek 6.17: Vyhledávání `KOAS25`



Obrázek 6.18: Vyhledávání `SA123AS`

Vstup může být zadán s libovolným množstvím "bílých znaků", tyto znaky ovšem nemají vliv na výsledek vyhledávání a jsou ignorovány. Vstup také může být zadán v libovolné kombinaci velkých, či malých písmen, opět bez vlivu na výsledek vyhledávání. Příklad je na obrázku 6.19



Obrázek 6.19: Vstup s mezerami

6.1.6 Shrnutí

Aplikace je schopná pracovat s vytvořenou databází a na základě vstupů poskytovat odpovídající výstupy a to i pro nekorektní vstupy. Uživatelé pracující s aplikací v rámci jejího testování neshledali žádný případ chybného chování. Následují souhrnné informace o zdrojovém kódu:

programovací jazyk	Python 2.7
řádek programu (včetně komentářů)	422
počet funkcí	12
velikost zdrojového souboru	12.6 kB

Tabulka 6.1: Informace o zdrojovém kódu aplikace pro vyhledvání

Výčet použitých knihoven a důvodů jejich použití je následující:

sys práce se soubory
os navigace v souborovém systému
re použití regulárních výrazů
ast použití funkce `literal_eval()` pro načtení slovníků speciálních znaků
json načtení permanentních proměnných
PIL práce se soubory obrázků
Tkinter tvorba GUI aplikace

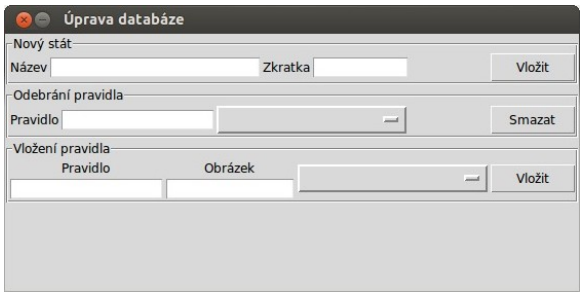
6.2 Aplikace pro editaci databáze RZ

Soubory obsahující pravidla v databázi jsou upravitelné standardním textovým editorem, neznalost pochodů v databázi však v takovém případě vede na nekonzistenci databáze, proti které je zavedeno několik ochranných prvků (viz 6.1.2). Pro úpravy, které korektně mění obsah databáze je vyvinutá aplikace editoru s GUI.

Úpravy proveditelné pomocí editoru jsou následující: přidání a odebrání pravidla z konkrétního státu a přidání nového státu do databáze.

6.2.1 Program

Aplikace je stejně jako aplikace vyhledávání vyvinuta v programovacím jazyku Python 2.7 s použitím knihovny *Tkinter* pro grafické uživatelské rozhraní. Ukázka vzhledu editoru viz obrázek 6.20 Prostředí aplikace obsahuje prvky knihovny *Tkinter*, představující vstupní pole, potvrzující tlačítka a nabídky států pro zamezení editace neexistujících států.



Obrázek 6.20: Editor databáze

Následují popisy a algoritmy jednotlivých případů použití editoru:

Přidání nového státu

Z popisu struktury databáze v 6.1.2 vyplývá, že stát je představován složkou, jejímž jménem je zkratka příslušného státu. Spojení této zkratky a jména státu je provedeno pomocí JSON souboru `statesdict.json`.

Je tedy třeba vytvořit složku příslušného státu a aktualizovat příslušný soubor s údaji o databázi. Tyto činnosti provádí funkce `crstate`. Vstup v podobě zkratky státu a jména státu je získán protřednitvím dvou objektů třídy `Entry`.

Zároveň se provádí i aktualizace `optionmenu`, což jsou výběrové nabídky pro výběr státu při přidávání a odebírání pravidla viditelné na obrázku 6.20. Účelem této aktualizace je možnost přidávání pravidel pro nově vytvořený stát bez nutnosti restartu aplikace.

Přidání nového pravidla

Úkoly nutné k přidání nového pravidla do databáze jsou následující: přidání pravidla do originálního souboru, přidání pravidla do `plates` souboru (viz 6.1.2), vložení propojovacího řádku do `imgs` souboru a případné vytvoření složky pro novou délku pravidel příslušného státu a aktualizace souboru `lens.json`. Příslušný algoritmus je znázorněn algoritmem 2 a implementován funkcí `add_rule`. Vstupy jsou získány objektů třídy `Entry`, představujících pravidlo a jméno korespondujícího obrázku a z objektu třídy `Optionmenu`, která umožňuje výběr státu.

Algorithm 2 Přidání pravidla do databáze

Require: zkratka státu `sc`, nové pravidlo `rule`, jméno obrázku `img`
  if neexistuje složka pro danou délku pravidla **then**
    vytvoř příslušnou složku
    vytvoř originální soubor
    vytvoř `imgs` soubor
    aktualizuj soubor `lens.json` o novou podporovanou délku státu `sc`
  end if
  $list1$ = řádky originálního souboru
  \\ seznam obsahující řádky originálního souboru
  if rule $\notin$ list1 **then**
    \\ vložení pravidla do originálního souboru
    přidej $rule$ do seznamu $list1$
    zapiš $list1$ do originálního souboru
  else

    \\ nejsou nutné úpravy
    `return`
  end if
  $list2$ = rotace(list1)
  \\ vytvoření tabulky pravidel
  str = prázdný řetězec
  for i $\in$ list2 **do**
    \\ přidávání řádků tabulky do řetězce
    $str = str + i$
  end for
  if `plates` soubor obsahuje slovník speciálních znaků **then**
    \\ přidání původních speciálních souborů
    dic = slovník speciálních znaků
    $str = dic + str$
  end if
  str = počet prvků $list1$ + str
  zápis str do `plates` souboru
  f = propojovací soubor `imgs`
  $f = f + img$

Odebrání pravidla

Odebrání pravidla z databáze zahrnuje tyto úkony: úprava originálního a `plates` souboru, vyjmutí odkazu příslušného obrázku s `imgs` souboru a případné vymazání celé složky délky pravidel a úprava souboru `lens.json`. Algoritmus je zachycen algoritmem 3 a vykonáván funkcí `del_rule`. Vstupy funkce jsou opět získány z GUI aplikace.

Algorithm 3 Odebrání pravidla z databáze

Require: zkratka státu `sc`, pravidlo `rule`
  if nexistuje složka pro danou délku pravidla **then return**
    \\ není co vymazat
  end if
  *list*1 = řádky originálního souboru
  \\ seznam obsahující řádky originálního souboru
  if rule ∈ list1 **then**
    \\ odebrání pravidla ze souboru
    vymaž *rule* z *list*1
    ulož *list*1 jako originální soubor
  end if
  if list1 je prázdný **then**
    \\ odebrané pravidlo bylo jediné v souboru
    smaž složku pro odpovídající délku pravidla
    aktualizuj soubor `lens.json` **return**
  end if
  l = řádky `imgs` souboru
  \\ seznam odkazů na obrázky
  vymaž *l*[*indexrulev list*1]
  \\ vymazání odpovídajícího odkazu
  *list*2 = rotace(list1)
  \\ vytvoření tabulky pravidel
  str = prázdný řetězec
  for i ∈ list2 **do**
    \\ přidávání řádků tabulky do řetězce
    str = *str* + *i*
  end for
  if `plates` soubor obsahuje slovník speciálních znaků **then**
    \\ přidání původních speciálních souborů
    dic = slovník speciálních znaků
    str = *dic* + *str*
  end if
  str = počet prvků *list*1 + *str*
  zápis *str* do `plates` souboru

6.2.2 Shrnutí a popis použití aplikace

Aplikace je schopná pracovat s vytvořenou databází a provádět její úpravy korektně z hlediska zachování její konzistence. Aplikace nepodporuje vkládání nových obrázků do aplikace, je předpokládáno, že takové zásahy bude provádět správce obeznámený s vnitřními principy databáze. Ze stejného důvodu aplikace neumožňuje uživateli smazat jakýkoliv stát.

Použití aplikace je triviální, uživatel zadá do příslušných vstupních polí nové údaje a stiskne příslušné tlačítko, aplikace kvůli bezpečnostním opatřením z důvodů nechtěného stisku klávesy *Enter* na tuto klávesu nereaguje. Veškeré změny jsou provedeny bez zásahu uživatele a jejich provedení je indikováno aktualizací GUI aplikace.

Aplikace byla testována sadou automatických testů, hlavní důraz byl kladen na korektní práci aplikace se soubory databáze a zachování konzistence databáze. Aplikace ovšem neumožňuje zjistit předchozí konzistenci databáze, proto je doporučeno provádět úpravy pouze pomocí tohoto editoru. Následují souhrnné informace o zdrojovém kódu:

programovací jazyk	Python 2.7
řádek programu (včetně komentářů)	410
počet funkcí	4
velikost zdrojového souboru	10.3 kB

Tabulka 6.2: Informace o zdrojovém kódu aplikace editoru

Výčet použitých knihoven a důvodů jejich použití je následující:

sys práce se soubory
os navigace v souborovém systému
shutil pokročilé operace s adresářovým stromem
json načtení a uložení permanentních proměnných
Tkinter tvorba GUI aplikace

V této kapitole byly představeny a demonstrovány obě vyvinuté aplikace spolu s algoritmy představujícími jejich fungování a porovnáním s existujícími aplikacemi. Byla popsána struktura databáze pravidel pro registrační značky a způsob, jak aplikace s touto databází pracují. Obě aplikace byly prokázány jako funkční a současně bylo uvedeno, že aplikace byly při vývoji testovány koncovými uživateli a sadami automatických testů. Obě aplikace byly implementovány v jazyce *Python 2.7* s použitím knihovny *Tkinter* pro vytvoření grafického uživatelského prostředí.

Kapitola 7

Závěr

V této práci byla vysvětlena teorie dvoudimensiálních formálních jazyků a automatů, které tyto jazyky dokážou analyzovat. Toto odvětví teoretické informatiky lze využít k popisu dvoudimensionálních objektů, tedy tabulek a obrazů. Takové objekty pak lze analyzovat danými automaty pro získání infromace o náležitosti konkrétního obrazu do jazyků. Jak bylo zmíněno, tyto poznatky lze využít například pro biometrické zabezpečení a analýzu obrazu.

Bylo vyvinuto několik automatů, jež umožňují analyzovat tabulky z hlediska výskytu konkrétního sloupce a tedy náležitosti takových tabulek do vytvořeného jazyka. Pomocí testování byl zjištěn nejvhodnější kandidát pro implementaci v rámci aplikace k vyhledávání státní příslušnosti registračních značek.

Registrační značky byly představeny pro několik evropských států, byla zmíněna různá pravidla těchto států a výjimky vážící se k těmto pravidlům. V analyzovaných státech byly rozebrány i velmi raritní tvary registračních značek pro úplnost databáze. Největší úkol v rámci praktické části práce bylo vytvoření databáze pravidel těchto značek. V současné době databáze čítá 27 podporovaných států a přibližně 9 000 pravidel.

V rámci práce byly vyvinuty dvě aplikace: aplikace původně zamýšlená, tedy aplikace pro vyhledávání v databázi pravidel registračních značek, a aplikace, která umožňuje editování této databáze. V práci byly zmíněny stěžejní algoritmy těchto aplikací a porovnání s aplikacemi, které se zabývají podobnou problematikou. Aplikace pro editaci umožňuje přidávání a odstranění pravidel bez nutné znalosti struktury databáze, mohou ji tedy využívat i koncoví uživatelé při menších obměnách pravidel značení vozidel.

Aplikace byly testovány koncovými uživateli, přičemž hlavním účelem tohoto testování bylo zjistit intuitivnost ovládání aplikací a důsledek jejich používání v rámci zefektivnění práce uživatelů. Obě aplikace jsou spustitelné na operačních sytémech UNIX a Windows. Pro použití v systémech UNIX je použit spustitelný zdrojový kód aplikace. Pro použití na systémech Windows byl pomocí nástroje *py2exe* vytvořen spustitelný program s příponou `exe`.

V současné době je hlavním záměrem ve vývoji aplikace zdokonalení a zvětšení databáze, případně úprava aplikace pro dávkové zpracování dat a její začlenění do většího celku pro vyhodnocování registračních značek. Výsledkem porovnání aplikace s existujícími aplikacemi je zjištění, že ve volně dostupných aplikacích neexistuje žádná, která by řešila stejný problém. Proto ji hodnotíme jako společensky přínosnou, jelikož umožňuje řešit aktuálně nezpracovanou problematiku.

Jako společensky přínosná byla práce označena i účastníky konference *Excel@FIT 2015*. Práce byla zahrnuta jako příspěvek do sborníku konference pod názvem *Identifikace státní*

příslušnosti vozidel pomocí dvoudimensionálních automatů v kategoriích *Zpracování dat* a *Překladače a gramatiky*. Zvláště dobře byla práce hodnocena akademickými pracovníky hostitelské fakulty za použití pokročilé teorie pro praktické využití.

Z hlediska vícedimensionálních automatů a jazyků je záměrem autorů do budoucna práce na třídimensionální obdobě těchto pojmů, která bude sloužit pro popis a klasifikaci třírozměrných těles. Tyto automaty a jazyky budou vhodné pro použití v modelovacích systémech.

Příklady použití mohou být definice složení těles z různých materiálů a klasifikace takových těles třidimensionálními automaty podle složení či tvaru. Stejně jako dvoudimensionální automaty dovolují detekci čtvercových objektů v obrazu, třídimensionální automaty by pak mohly umožnit detekci krychlových těles v prostoru a při vyšší abstrakci i detekci těles složitějších. Operace konkatenace a rotace pak dostávají nový rozměr z hlediska více možných os rotace a mnoha různých směrů konkatenace dvou těles. Pomocí těchto operací by pak bylo možné definovat složitější tělesa vzniklá konkatenací elementárních těles. Takový popis těles je použit některými modelovacími editory. Jako zajímavá se pak jeví definice tzv. *formálních jazyků těles*, tedy těles, která by mohla být popsána speciálně vyvinutými regulárními výrazy pro tělesa.

Literatura

[1] Rozporzšdzenie Ministra Infrastruktury w sprawie rejestracji i oznaczania pojazdów. 2002.
URL http://isap.sejm.gov.pl/DetailsServlet?id=WDU20021331123

[2] Vzory tabuliek s evidenčným číslom pridelované na vozidlá, Ministerstvo vnútra SR - Polícia. 2007.
URL http://www.minv.sk/?vzory-tabuliek-s-evidencnym-cislom-pridelovane-na-vozidla

[3] Vyhláška č. 343/2014 Sb., o registraci vozidel. 2014.
URL http://www.psp.cz/sqw/sbirka.sqw?cz=343&r=2014

[4] Blum, M.; Hewitt, C.: Automata on a 2-dimensional tape. In *Switching and Automata Theory, 1967. SWAT 1967. IEEE Conference Record of the Eighth Annual Symposium on*, IEEE Publishing, 1967, s. 155–160.
URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5397208

[5] Dong, J.; Jin, W.: Comparison of Two-Way Two-Dimensional Finite Automata and Three-Way Two-Dimensional Finite Automata. In *Computer Science & Service System (CSSS), 2012 International Conference on*, IEEE, 2012, ISBN 9781467307215, s. 1904–1906.
URL http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6394793

[6] Florczyk, S.: OCR. In *Robot Vision*, Weinheim, FRG: Wiley-VCH Verlag GmbH, 2005, ISBN 9783527405442, s. 129–131.

[7] Giammarresi, D.: Two-Dimensional Languages and Recognizable Functions. In *Developments in Language Theory*, 1993, s. 290–301.

[8] Inoue, K.; Nakamura, A.: Two-dimensional multipass on-line tessellation acceptors. *Information and Control*, ročník vol. 41, č. issue 3, 1979: s. 305–323.
URL http://linkinghub.elsevier.com/retrieve/pii/S0019995879906041

[9] Inoue, K.; Takanami, I.; Nakamura, A.: A note on two-dimensional finite automata. *Information Processing Letters*, ročník 7, č. 1, 1978: s. 49 – 52, ISSN 0020-0190.
URL http://www.sciencedirect.com/science/article/pii/0020019078900406

[10] Krajíček, P.: Rozpoznání SPZ/RZ. 2010.

[11] Marinov, P.: *České poznávací značky*. Praha: Kampe, první vydání, c2007, ISBN 978-80-902955-7-5.

[12] Meduna, A.: *Automata and languages*. London: Springer, 2000, ISBN 1852330740.

[13] Rovný, J.; Bolcha, P.; Babin, J.: Přidělují SPZ pražští úředníci náhodně? 2012. URL http://isis.vse.cz/zp/94349

[14] Rozenberg, G.; Salomaa, A.: *Handbook of formal languages / vol. III, Beyond words*. Berlin: Springer Verlag, 1997, ISBN 3540606491.

[15] Smékal, D.: Detekce státní poznávací značky pro dohledové systémy. 2013.

Seznam příloh

- Příloha A — obsah přiloženého CD
- Příloha B — vzory registračních značek

Příloha A

Obsah CD

- Tato práce ve formátu Portable Document Format – `thesis.pdf`
- Složka `tex` se zdrojovými kódy pro vytvoření této práce ve formátu LaTeX
- Složka `project` se soubory programů a databáze
  - Složka `apps`
    - Složka `src` – zdrojové kódy programů (search.py, editor.py)
    - Složka `bin` – binární spustitelné soubory programů generované pomocí *py2exe*
    - Složka `conf` – konfigurační JSON soubory
  - Složka `plates` – databáze pravidel značek
  - Složka `imgs` – databáze obrázků vlajek a značek
- Manuál pro programy – `manual.pdf`

Příloha B

Vzory registračních značek

V této příloze jsou znázorněny různé tvary RZ podporovaných aplikací pro demonstraci rozmanitosti možných grafických řešení a tvarů textu.

Obrázky, u kterých není uveden zdroj byly převzaty z http://www.olavsplates.com/, což je zájmová internetová databáze fotografií světových RZ.

B.1 Česká republika

Tento stát byl analyzován v rámci sekce 5.2.1.

Obrázek B.1: Standardní tvar od 1967 do 2001

Obrázek B.2: Standardní tvar od 2001

Obrázek B.3: Standardní tvar od 2001 se sériovým písmenem

Obrázek B.4: Značka pro osoby s imunitou

Obrázek B.5: Značka vozidla pro export

B.2 Slovensko

Tyto značky byly analyzovány v rámci sekce 5.2.2. Zdrojem obrázků pro tento stát byla stránka [2].

Obrázek B.6: Standardní slovenská RZ

Obrázek B.7: Značka pro vozidlo policie

Obrázek B.8: Značka pro zkušební vozidlo

Obrázek B.9: Značka pro historické vozidlo

Obrázek B.10: Značka armádního vozidla

Obrázek B.11: Značka diplomatického vozidla

B.3 Polsko

Tyto značky byly analyzovány v sekci 5.2.3. Je zobrazen pouze zlomek tvarů, které byly uvedeny, z důvodu velké obsáhlosti tohoto značení, jak bylo zmíněno v příslušné sekci.

Obrázek B.12: Standardní tvar značení 1

Obrázek B.13: Standardní tvar značení 2

Obrázek B.14: Standardní tvar značení 3

Obrázek B.15: Diplomatická značka vozidla

Obrázek B.16: Značka pro vozidla v procesu testování

Obrázek B.17: Značka pro historická vozidla (zdroj: http://polskietablice.fm.interiowo.pl)

B.4 Německo

Tvary značení tohoto státu byly analyzovány v sekci 5.2.4. Jedná se o stát s nejvíce pravidly v databázi díky velké variabilitě pravidel.

Obrázek B.18: Standardní tvar značení 1

Obrázek B.19: Standardní tvar značení 2

Obrázek B.20: Značka s dočasnou platností

Obrázek B.21: Značka vozidla pro prodej a převoz

B.5 Rakousko

Tyto značky jsou analyzovány v sekci 5.2.5. Jak je řečeno v této sekci, tento stát má velmi volná pravidla značení, umožňující stovky kombinací.

Obrázek B.22: Tvar standardních značek z menších regionů

Obrázek B.23: Standardní značka registrovaná ve Vídni

Obrázek B.24: Dočasná rakouská značka

Obrázek B.25: Značka pro rakouské diplomaty

B.6 Belgie

Tento stát používá výlučně registrační značky s červeným textem, což snižuje jeho čitelnost na záznamech. Navzdory tomu, že Belgie je jeden ze zakladatelských států Evropské unie, patří zároveň mezi státy, které jako poslední přijali tvar RZ s *euroštítkem*. Typický tvar dnes stále častý na silnicích je na obrázku B.28, tento tvar povolovoal libovolnou kombinaci 3 písmen a 3 číslic.

Obrázek B.26: Nový belgický standardní tvar

Obrázek B.27: Tvar belgické diplomatické značky

Obrázek B.28: Starší tvar belgické značky

B.7 Velká Británie

Tento stát stále vydává značení bez *euroštítku*, který je pouze volitelným doplňkem. Ovšem značky tohoto státu jsou viditelně odlišné od ostatních díky speciálnímu typu písma. Zajímavostí je, že z čísla na RZ lze určit rok vydání RZ podle jistého vzorce.

Obrázek B.29: Standardní anglická značka

Obrázek B.30: Standardní anglické značka s voliteným euroštítkem

B.8 Řecko

Řecko jako stát se speciální abecedou pužívá pouze několik písmen abecedy pro registrační značky, jedná se o znaky, jenž mají stejný vizuální vzhled jako písmena latinky. To však platí jen pro standardní značení, například diplomatická vozidla mohou obsahovat i písmena ostatní. Současná standardní řecká RZ je vizuláně velmi podobná staršímu tvaru českých RZ.

Obrázek B.31: Řecká RZ s euroštítkem

Obrázek B.32: Standardní řecká značka

Obrázek B.33: Řecká diplomatická značka s písmeny sigma a delta

Obrázek B.34: Řecká značka pro rodiny s více než 4 dětmi (osvobozeno od daní)

B.9 Nizozemsko

Tento stát povoluje jakoukoliv kombinaci znaků, pokud je dodržen daný rozsah textu. Jednotlivé kombinace písmen a číslic jsou uvolňovány během let pro navýšení kapacity systému.

Obrázek B.35: Standardní tvar holandské RZ 1

Obrázek B.36: Standardní tvar holandské RZ 2

Obrázek B.37: Holandská diplomatická značka

B.10 Ruská federace

Rusko používá abecedu azbuky, pro koexistenci s evropskými značkami jsou tedy použity pouze některá písmena, která mají vizuální obdobu mezi standardními znaky. Nutno podotknout, že například znak C značí v azbuce písmeno S, to však není problémem pro aplikaci. Každý tvar obsahuje dodatečný kód oblasti na pravé straně RZ.

Obrázek B.38: Standardní ruská značka

Obrázek B.39: Ruská diplomatická značka

B.11 Turecko

Turecké RZ se vydávají ve dvou sériích, jde o RZ se znaky latinky a RZ se znaky arabské abecedy, budeme se tedy zajímat pouze o první sérii. Turecké speciální RZ jsou odlišeny pouze barvou podkladu písma. Všechny tvary jsou předcházeny dvojciferným číslem oblasti (01 až 81).

Obrázek B.40: Turecká standardní značka 1

Obrázek B.41: Turecká standardní značka 2