VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

DVOUDIMENSIONÁLNÍ VERZE SKÁKAJÍCÍCH AUTOMATŮ A JEJICH APLIKACE

TWO-DIMENSIONAL VERSIONS OF JUMPING AUTOMATA AND THEIR APPLICATIONS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. DOMINIK ŠVAČ**
AUTHOR

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

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2021/2022

Zadání diplomové práce

Student: **Švač Dominik, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Počítačová grafika a interakce
Název: **Dvoudimensionální verze skákajících automatů a jejich aplikace**
Two-Dimensional Versions of Jumping Automata and Their Applications
Kategorie: Teoretická informatika

Zadání:

1. Seznamte se detailně s různými variantami skákajících a dvoudimensionálních automatů.
2. Zaveďte dvoudimensionální verze skákajících automatů dle instrukcí školitele.
3. Studujte vlastnosti automatů zavedených v bodě 2 dle instrukcí školitele.
4. Implementujte automaty zavedené v bodě 2 pro zpracování obrazu.
5. Demonstrujte aplikaci automatů navržených v bodě 2 např. v kontextu obrazců obsahující čísla a operace s nimi. Implementujte tuto aplikaci a testujte ji.
6. Zhodnoťte dosažené výsledky a diskutujte další možný vývoj projektu.

Literatura:

- Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages, Volume 1-3, Springer, 1997, ISBN 3-540-60649-1
- Horáček, P., Meduna, A., Tomko M.: Handbook of Mathematical Models for Languages and Computation, IET, 2020, ISBN 978-1-78561-659-4.
- Švač, D.: Dvoudimensionální konečné automaty a jejich aplikace, bakalářská práce, FIT VUT v Brně, 2020

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Meduna Alexander, prof. RNDr., CSc.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 18. května 2022
Datum schválení: 26. října 2021

Abstrakt

Cieľom tejto práce bolo vytvoriť typ automatu, ktorý umožňuje rozpoznávanie objektov zo vstupného obrázka. Tvorený je z dvoch typov automatov, dvojrozmerný konečný automat a skákajúci automat. Aplikácia tohoto automatu umožňuje nájsť čísla v obrázku. Rozpoznávanie prebieha pomocou rozhodovacieho stromu, ktorý pomáha urýchliť nájdenie správneho čísla.

Abstract

The aim of this work was to create a type of automaton that allows the recognition of objects from the input image. It consists of two types of automaton, a two-dimensional finite automata and jumping automata. The application of this automata allows you to find the numbers in the image. Recognition is using a decision tree, which helps speed up finding the right number.

Kľúčové slová

automat, konečný automat, dvojrozmerný automat, skákajúci automat, rozhodovací strom, rozpoznávanie čísel

Keywords

automaton, finite automaton, two - dimensional automaton, jumping automaton, decision tree, number recognition

Citácia

ŠVAČ, Dominik. *Dvoudimensionální verze skákajících automatů a jejich aplikace*. Brno, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. RNDr. Alexander Meduna, CSc.

Dvoudimensionální verze skákajících automatů a jejich aplikace

Prehlásenie

Prohlašuji, že jsem tuto diplomovú práci vypracoval samostatně pod vedením pana prof. RNDr. Alexander Meduna,CSc. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Dominik Švač
10. mája 2022

Poďakovanie

Veľmi rád by som poďakoval pánovi prof. RNDr. Alexandrovi Medunovi,CSc za cenné rady, pomoc a poskytnutý čas, ktorý mi venoval pri riešení tejto práce.

Obsah

Kapitola 1

Úvod

V tejto práci sa čitateľ dozvie, ako fungujú vybrané typy konečných automatov. Každý typ je iný. Líšia sa hlavne spôsobmi ako prechádzajú po vstupnej páske. Opísané sú tu dvojrozmerné automaty, ktoré sa spojili zo skákajúcimi a vznikol tak nový typ. Následne bude stručný úvod ku rozhodovacím stromom a na záver práce je opísaná implementácia a testovanie výslednej aplikácie.

Klasické deterministické automaty prechádzajú postupne, od začiatku po koniec a čítajú symbol po symbole. Dvojcestné automaty umožňujú pohyb aj dozadu. Môže sa zdať, že takéto automaty majú väčšiu vyjadrovaciu silu ako klasické, ktoré sa pohybujú iba jedným smerom. Avšak v skutočnosti sú rovnaké. Nasledujú skákajúce automaty, ktoré sú pomerne nové. Tie majú definovaný špeciálny pohyb po páske. Nemusia ísť sekvenčne, môžu sa posunúť na ľubovoľný symbol v hocijakom smere. Sú to takzvané skoky.

Pokračuje sa automatmi, ktoré už nemajú vstupnú pásku jednorozmernú. Teoreticky môže byť n-dimenzionálna. Pri väčšine prípadov sa jedná o dvojrozmernú. Tá môže byť napríklad obrázok, v ktorom sa bude chodiť na základe určitých pravidiel. Opísané sú tu ich rôzne varianty. Štvorcestné automaty využívajú na presun čítacej hlavice štyri základné pohyby do všetkých smerov a to hore, dole, doprava a doľava. On-line Tesselačné automaty majú odlišný spôsob pohybu. V jednom kroku sa nemusí čítať iba jeden symbol. Pohybujú sa v takzvaných prechodových vlnách. Bunky (jednotlivé časti vstupnej pásky) menia svoj stav na základe okolitých buniek.

Existuje veľa takýchto automatov,ktoré sú zaujímavé, ale do tejto práce sa nedostali. Napríklad varianta celulárnych automatov, hra Game of Life. Má definovaných pár jednoduchých pravidiel. Ako vstup má štvorcovú bunkovú mriežku. Na začiatku sa zvolí počiatočná konfigurácia a následne hra beží sama pomocou pravidiel. V priebehu času vznikajú rôzne zaujímavé útvary. Pri niektorých konfiguráciách sa skončí hneď po pár krokoch, iné sa vykonávajú do nekonečna.

Ďalšia časť je zameraná na nový typ dvojdimenzionálneho automatu, ktorý má viac možností ako prechádzať po vstupnej páske. Podobne ako štvorcestný automat dokáže chodiť do všetkých smerov (hore, dole, doprava, doľava). Zároveň má však definované aj skoky. Tým pádom môže z ľubovoľného miesta na vstupnej páske skočiť na iné miesto a odtiaľ sa pohybovať ďalej. Táto možnosť tak umožňuje oveľa rýchlejšie rozpoznávanie v prípade, že sa napríklad dopredu vie, kde sa nachádza určitý objekt, ktorý chceme rozpoznať. Klasické automaty by museli postupne prejsť páskou a dostať sa do konkrétneho bodu. Dvojdimenzionálne skákajúce automaty môžu takmer okamžite skočiť do dopredu určeného miesta.

Takisto aj grafická reprezentácia vo forme automatov by bola zložitejšia bez skokov. Skoky zmenšujú počet stavov a tým sa vo veľa prípadoch graf zjednoduší a sprehľadní.

Pri veľkej vstupnej páske (napríklad obrázok s vysokým rozlíšením) sa môže vykonať optimalizácia a urýchli sa tak celý proces priebehu automatu. V prípade, že sa prehľadáva páska sekvenčne, môžu sa riadky preskakovať. Nemusí sa tak kontrolovať každý jeden riadok. Napríklad sa bude kontrolovať každý druhý, poprípade tretí riadok. Táto optimalizácia ale môže niekdy dospieť k nepresným výsledkom rozpoznania, keďže sa tak stráca časť informácii.

Využitím tohto typu automatu sa v ďalšej časti realizuje jeho využitie. Zameraná je na rozpoznávanie čísel v obrázku. Pre najlepšie výsledky je vhodné mať veľký kontrast medzi číslom a jeho okolím, Napríklad čierne čísla na bielom podklade, alebo opačne. V aplikácii sa dá nastaviť vyhľadanie prvého výskytu, alebo všetkých. Vstupom môže byť konkrétny obrázok, ale aj zložka. V takom prípade sa v nej automaticky vyhľadajú všetky podporované obrázky a výsledkom bude jednoduchý report.

Implementovaných je niekoľko automatov. Dokážu rozoznať rôzne typy čiar, ako horizontálna, vertikálna, alebo šikmá. Rozpoznávanie prebieha na základe rozhodovacieho stromu, ktorý určuje, ako sa bude číslo vyhodnocovať. Každý uzol obsahuje informáciu o tom, ktorý automat sa spustí, kde sa presunie čítacia hlavica, poprípade či sa spustí podmienka, ktorá porovná určité hodnoty. Najprv sa kontroluje ľavá časť čísla a potom pravá.

V poslednej kapitole sa nachádza testovanie výslednej aplikácie. Testy sa vykonávali prevažne na upravenom datasete MINIST. Je to dataset písaných čísel. Najviac sa používa v oblasti neurónových sieti na trénovanie a testovanie.

V aplikácii sa nachádza veľa rôznych premenných, ktoré sa dajú testovať. Najčastejšie ide o presnosť alebo rýchlosť nájdenia výsledného čísla. Napríklad sa dajú testovať rôzne varianty automatov na sekvenčné prehľadávanie vstupu a hľadanie začiatku čísla. Jednou z možností je preskakovanie riadkov, poprípade stĺpcov obrázka. Môžu sa využiť aj rôzne paralelné metódy. Možností je mnoho.

Jedným z dôležitých faktorov je aj prahová hodnota, ktorá kontroluje, či je daný pixel biely, alebo čierny, poprípade či patrí do okolia alebo je súčasťou čísla. Preto jeden z testov je zameraný práve tento faktor. Taktiež sa tu nachádzajú grafy a tabuľky, ktoré znázorňujú, ako rýchlo prebieha rozpoznanie jednotlivých čísel.

Okrem existujúceho bude na testovanie použitý aj vlastný dataset. Obsahuje 179 obrázkov, na ktorých sú čísla. Je to síce menej, ako pri upravenom datasete MNIST, ale každý jeden bol ručne odfotený. Majú rôzne veľkosti. Väčšinou boli písané čiernym, alebo tmavomodrým perom na svetlom pozadí (biely papier).

Kapitola 2

Existujúce formy konečných automatov

Táto kapitola je zameraná na teoretický úvod k automatom. Čitateľ tu nájde niektoré základné pojmy a definície o konečných automatoch. Najprv sú tu opísané klasické konečné, deterministické aj nedeterministické automaty. Nasledujú skákajúce automaty. Tie sa pohybujú po páske a v ľubovoľnom mieste môžu preskočiť na iný symbol. To im umožňuje mať väčšiu vyjadrovaciu silu.

Opísané sú tu aj niektoré známe druhy dvojrozmerných konečných automatov. Na vstup dostanú obrázok. Pomocou prechodových funkcií sa posúvajú, až kým nenarazia na koncový stav. Budeme predpokladať, že prechod z jedného stavu do druhého je okamžitý. V skutočnosti to ale vyžaduje určitý čas.

2.1 Konečné stavové automaty

Sú to základné stavebné prostriedky, ktoré nám umožňujú popis regulárnych jazykov. Máme dva základné modely a to deterministické konečné automaty (DFA) 2.1.2 a nedeterministické konečné automaty (NFA) 2.1.2. Sú takmer totožné, ale rozdiel je v ich prechodovej funkcii. Vyjadrovacia sila oboch modelov je rovnaká. Každý NKA sa dá tým pádom previesť na ekvivalentný DKA.

2.1.1 Základné pojmy

Abeceda

Označuje sa Σ, je to konečná neprázdna množina symbolov [1]

- $\{0, 1\}$
- $\{a,b,c\}$
- $\{\langle 0, 0\rangle,\langle 0, 1\rangle,\langle 1, 0\rangle,\langle 1, 1\rangle\}$

Jazyk

Máme danú abecedu Σ a nech Σ^* značí množinu všetkých reťazcov nad abecedou Σ. Potom jazyk nad abecedou Σ je každá podmnožina $L \subseteq \Sigma^*$.

- L = Φ
- L = {w ∈ {0,1}* | |w| > 5}

Reťazec

Pre danú abecedu Σ a prirodzené číslo n platí:

- ϵ je reťazec nad abecedou Σ
- reťazec je postupnosť symbolov a_1a_2 ... a_n, pričom n je dĺžka reťazca a pre každé i = 1,2 ... n, $a_i \in \Sigma$

Epsilon

Symbol ϵ značí prázdny reťazec, teda taký, ktorý neobsahuje žiadny symbol

Dĺžka reťazca

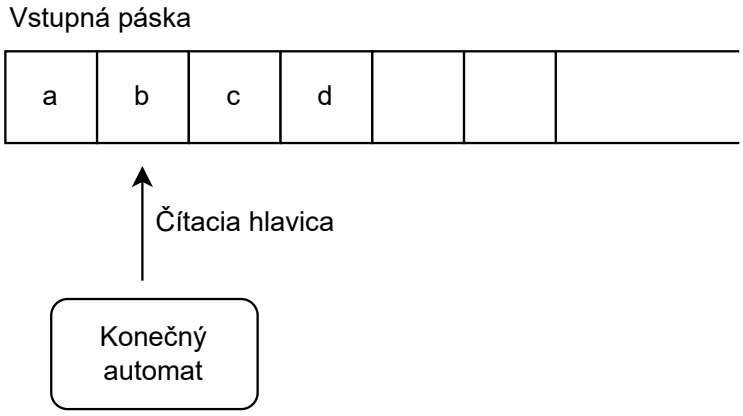
Pre danú abecedu Σ a reťazec x = a_1a_2 ... a_n nad Σ platí že |x| = |a_1a_2 ... a_n| = n

Konkatenácia

Máme abecedu Σ, a nech x = a_1a_2 ... a_n a y = b_1b_2 ... b_n sú reťazce nad Σ, konkatenácia dvoch reťazcov xy = a_1a_2 ... $a_nb_1b_2$ ... b_n

2.1.2 Deterministický konečný automat

Deterministický konečný automat [1] (anglicky: Deterministic finit automaton DFA). Je to matematický model stroja, ktorý akceptuje konkrétnu množinu reťazcov nad abecedou Σ. Obsahuje čítaciu hlavicu, ktorá číta dáta zo vstupnej pásky. Pohybuje sa smerom na koniec pásky.



Obr. 2.1: Konečný automat

Definícia

DFA je pätica (Q,Σ,δ,q_0,F),ktorá pozostáva z nasledujúcich častí: [1]

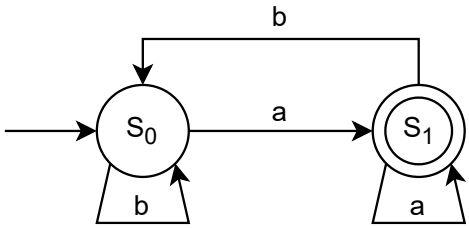
- Konečná množina stavov, často označovaná ako Q
- Konečná množina vstupných symbolov, často označovaná ako Σ
- Prechodová funkcia (Transition function), ktorá berie ako argumenty stav a vstupný symbol a výsledkom je ďalší stav. Prechodná funkcia je najčastejšie označovaná ako δ.
- Počiatočný stav q_0, ktorý je z množiny Q.
- Množina koncových stavov alebo prijatých stavov, označovaná ako F. Kde F je podmnožina Q.

Príklad

Máme automat DFA, ktorý prijíma reťazce, ktoré obsahujú iba symboly a, b a reťazec musí končiť symbolom a. Formálne môžeme zapísať tento jazyk ako:

L = { w | w = xa, x môže obsahovať iba a a b}

Obsahuje stavy S_0, S_1, kde S_0 je počiatočný stav a S_1 je koncový stav. [2]



Obr. 2.2: Konečný automat prijímajúci jazyk L

Diagram prechodov

Diagram prechodov (Transition diagram) [2] je graf, ktorý je definovaný nasledovne:

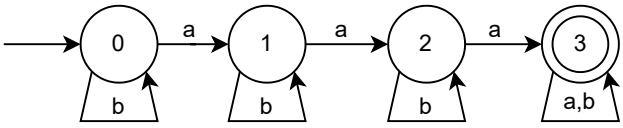
- Pre každý stav z množiny Q existuje uzol.
- Každý stav q patrí do množiny Q, a každý vstupný symbol a patrí do množiny Σ. Existuje prechodová funkcia δ(q,a) = p. Potom existuje prechod z uzla q do uzla p označený ako a. Ak existuje viac vstupných symbolov, ktoré vytvárajú prechod z q do p, potom má prechodový diagram jeden prechod označený zoznamom všetkých týchto symbolov.
- Prechodový diagram obsahuje aj šípku, ktorá vchádza do počiatočného uzlu q_0. Je označená ako Štart a nevychádza zo žiadneho uzla.
- Koncové stavy sú označené dvojitým kruhom a sú z množiny F. Stavy, ktoré nie sú z množiny F, sa značia jedným kruhom.

Príklad 2

Definujme si automat M, ktorý obsahuje štyri stavy Q = {0,1,2,3}. Vstupná abeceda pozostáva zo symbolov $\Sigma = \{a, b\}$. Počiatočný stav je 0 a v množine koncových stavov je stav 3. Prechodová funkcia je definovaná nasledovne [3]:

$$\delta(0,a) = 1,$$
$$\delta(1,a) = 2,$$
$$\delta(2,a) = \delta(3,a) = 3,$$
$$\delta(q,b) = q,\ q \in \{0,1,2,3\}$$

Tento automat môžme špecifikovať aj pomocou tabuľky alebo prechodového diagramu



Obr. 2.3: Diagram prechodov

	a	b
→0	1	0
1	2	1
2	3	2
3F	3	3

Obr. 2.4: Tabuľka

2.1.3 Nedeterministický konečný automat

Nedeterministický konečný automat (anglicky: Nondeterministic finit automaton NFA). Podobne ako DFA aj NFA má konečnú množinu stavov, konečnú množinu symbolov, počiatočný stav a množinu koncových stavov. Rozdiel je v prechodovej funkcii δ.

Prechodová funkcia pre NFA má ako argumenty stav a vstupný symbol. Výsledkom môže byť jeden stav, žiaden, alebo aj viac stavov.

Definícia

NFA je pätica (Q,Σ,δ,q_0,F) kde: [1]

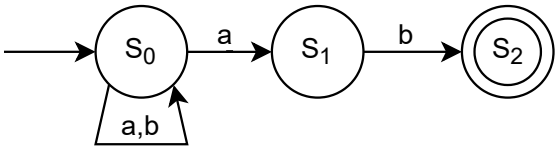
- Q je konečná množina stavov
- Σ je konečná množina vstupných symbolov
- q_0 je počiatočný stav, kde $q_0 \in$ Q
- F je konečná množina koncových stavov, pričom F ⊆ Q
- δ je prechodová funkcia, ktorá berie stav z Q a vstupný symbol z Σ ako vstupy a výsledkom je podmnožina množiny Q.

Príklad

Máme automat NFA, ktorý spracováva reťazce obsahujúce iba symboly a a b. Reťazec končí postupnosťou ab. Začína v stave S_0. Ak bude v tomto stave na vstupe a, nevieme presne určiť, či je to znak, ktorým sa začína koniec reťazca, alebo sa pokračuje ďalej. Preto sú v tomto stave dva prechody, jeden naspäť do seba a druhý do stavu S_1. Formálne môžme zapísať tento jazyk takto:

L = { w | w = xab, x môže obsahovať iba a a b}

Obsahuje stavy S_0, S_1 a S_2, kde S_0 je počiatočný stav a S_2 je koncový stav. [2]



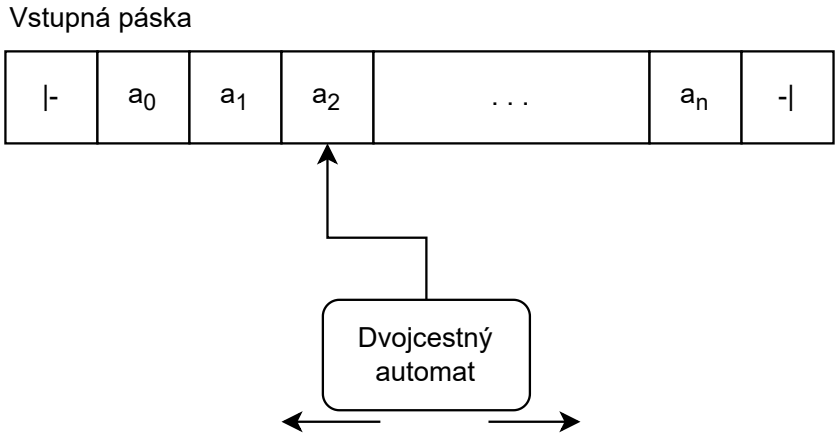
Obr. 2.5: Nedeterministický konečný automat

2.1.4 Dvojcestný konečný automat

Dvojcestné konečné automaty 2.1.4 (2DFA) sú podobné ako DFA, avšak s tým rozdielom, že môžu čítať vstupný symbol v oboch smeroch. Máme tu čítaciu hlavicu, ktorá sa môže pohybovať do prava alebo do ľava. Taktiež má svoju deterministickú (2DFA) aj nedeterministickú verziu (2NFA).

Môže sa zdať, že takéto automaty majú väčšiu vyjadrovaciu silu ako klasické, ktoré sa pohybujú iba jedným smerom. Avšak v skutočnosti sú rovnaké. Prijímajú iba jazyky z regulárnej množiny.

Vstupný reťazec je ohraničený z oboch strán symbolmi { |-, -| }, ktoré nie sú súčasťou vstupnej abecedy Σ. Čítacia hlava sa nemôže pohnúť mimo hraničných symbolov [3].



Obr. 2.6: Vstupná páska 2DFA

Definícia

Dvojcestné konečné automaty [3] (anglicky: Two-Way Finite Automata) sú definované nasledovne: M = (Q,Σ,|-,-|,δ, s,t,r) kde

- Q je konečná množina stavov
- Σ je Konečná množina vstupných symbolov
- |- je ľavý koncový symbol, pričom platí |- $\notin \Sigma$
- -| je pravý koncový symbol, pričom platí -| $\notin \Sigma$
- δ je prechodová funkcia v tvare: Q × (Σ ∪ { |-,-| }) → (Q × { L,R }), pričom L a R sú smery pohybu čítacej hlavice
- s je počiatočný stav, ktorý je z množiny Q.
- t je akceptujúci stav, ktorý je z množiny Q.
- r je neakceptujúci stav, ktorý je z množiny Q.

Príklad

Máme 2DFA automat M, ktorý pozostáva zo stavov Q = { $q_0, q_1, q_2, p_0, p_1, t, r$ } a vstupnej abecedy Σ = { a,b}. Počiatočný stav je q_0. Prechodová funkcia δ je daná nasledujúcou tabuľkou:

	$\vdash$	a	b	$\dashv$
q_0	(q_0, R)	(q_1, R)	(q_0, R)	(p_0, L)
q_1	–	(q_2, R)	(q_1, R)	(r, L)
q_2	–	(q_0, R)	(q_2, R)	(r, L)
p_0	(t, R)	(p_0, L)	(p_1, L)	–
p_1	(r, R)	(p_1, L)	(p_0, L)	–
t	(t, R)	(t, R)	(t, R)	(t, L)
r	(r, R)	(r, R)	(r, R)	(r, L)

Obr. 2.7: Tabuľka prechodovej funkcie [3]

Automat M prijíma nasledujúci jazyk:

L = { $x \in$ {a,b}* | #a(x) je násobkom 3 a #b(x) je párny }

2.2 Skákajúce automaty

Táto sekcia zobrazuje špeciálny typ automatov. Ako názov hovorí, povolené sú tu skoky na vstupnej páske. Takéto automaty obsahujú vstupnú pásku, čítaciu hlavicu a konečnú množinu pravidiel.

Vstupná páska je rozdelená do buniek. Každá bunka obsahuje jeden symbol vstupného reťazca. Symbol, na ktorý ukazuje čítacia hlavica, je aktuálny vstupný symbol. Pravidlá na základe aktuálneho stavu a práve čítaného symbolu zvolia stav, do ktorého sa automat dostane. Pri klasickom konečnom automate by sa posunula čítacia hlavica smerom do prava a prečítal by sa ďalší symbol na vstupnej páske. Prečíta tak reťatec z ľava do prava a rozhodne, či tento reťazec príjme, alebo nie. Pri skákajúcich automatoch sa môže čítacia hlavica posunúť na ľubovoľný symbol v hocijakom smere. Ak už bol raz symbol prečítaný, nemôže byť prečítaný znova.

2.2.1 Definície

Všeobecný skákajúci konečný automat

Všeobecný skákajúci konečný automat (anglicky: General jumping finit automaton GJFA). Je to pätica (Q,Σ,R,s,F). Pozostáva z nasledujúcich častí [5]:

- Konečná množina stavov, často označovaná ako Q
- Konečná množina vstupných symbolov, často označovaná ako Σ
- Množina pravidiel, ktoré sú v tvare: $py \to q$ ($p, q \in Q$, $y \in \Sigma^*$)
- Počiatočný stav s, ktorý je z množiny Q.
- Konečná množina koncových stavov alebo prijatých stavov, označovaná ako F. Kde F je podmnožina Q.

Táto definícia je takmer identická s klasickými konečnými automatmi 2.1.2, ale líši sa v pravidlách, kde môžu prebiehať skoky 2.2.1. To im dodáva väčšiu vyjadrovaciu silu ako DFA.

Skákajúci konečný automat

Nech máme GJFA M = (Q,Σ,R,s,F). Ak pravidlá M sú v tvare $py \to q \in$ R. Pre veľkosť symbolu $y \in \Sigma^*$ platí nasledovné: |y| $\leqq$ 1. [5]

Deterministický JFA

Nech máme GJFA M = (Q,Σ,R,s,F). M je deterministický ak platí pre pravidlá v tvare $py \to q \in$ R, kde |y| = 1. Pre každé $p \in Q$ a každé $a \in \Sigma$ nesmie byť viac ako jedno $q \in Q$ tak, že $pa \to q \in R$. [5]

Skok

Označuje sa najčastejšie symbolom $\curvearrowright$. Nech existuje $x, z, x', z' \in \Sigma^*$ tak, že $xz = x'z'$ a $py \to q \in$ R, potom automat M urobí skok z $xpyz$ do $x'qz'$. Zapisuje sa to ako $xpyz \curvearrowright x'qz'$. [5]

Ľavý a pravý skok

Máme automat M = (Q,Σ,R,s,F), ktorý je GJFA. Nech $w, x, y, z \in \Sigma^*$ a pravidlo v tvare $py \to q \in$ R, potom M môže vykonať ľavý skok z reťazca $wxpyz$ do $wqxz$. Značí sa to nasledovne: [5]

$$wxpyz \ {}_l\curvearrowright wqxz$$

pre pravý skok z reťazca $wpyxz$ do $wxqz$ automatu M sa značí nasledovne:

$$wpyxz \ {}_l\curvearrowright wxqz$$

Nech existujú $u, v \in \Sigma^*$ $Q\Sigma^*$, potom $u \curvearrowright v$ platí práve vtedy, ak $u \ {}_l\curvearrowright v$ alebo $u \ {}_r\curvearrowright v$. Množina jazykov pre pravé a ľavé skoky môže vyzerať nasledovne:

$${}_l\text{L(M,}\curvearrowright\text{)} = \{\text{u,v} \mid \text{u,v} \in \Sigma^*, \text{usv } {}_l\curvearrowright^* f \text{ kde } f \in F\}$$

$${}_r\text{L(M,}\curvearrowright\text{)} = \{\text{u,v} \mid \text{u,v} \in \Sigma^*, \text{usv } {}_r\curvearrowright^* f \text{ kde } f \in F\}$$

Jazyk prijímaný GJFA

Značí sa ako L(M,$\curvearrowright$) a je definovaný nasledovne: [5]

L(M,$\curvearrowright$) = { uv | u,v $\in \Sigma^*$, usv $\curvearrowright$ f, f $\in$ F}.

Vieme povedať, že M akceptuje w $\in \Sigma^*$ práve vtedy, ak w $\in$ L(M,$\curvearrowright$), inač M odmieta. Jazyky dvoch automatov GJFA M a GJFA M' su ekvivalentné ak L(M,$\curvearrowright$) = L(M',$\curvearrowright$).

2.2.2 Príklady

Príklad na JFA

Máme daný JFA M = (Q,Σ,R,s,F), kde:

- Q = $\{S_0, S_1, S_2\}$
- Σ = {0,1,2}
- R = $\{S_0 0 \to S_1,\ S_1 1 \to S_2,\ S_2 2 \to S_0\}$
- s = S_0
- F = $\{S_0\}$

Tento automat akceptuje tie jazyky kde je počet symbolov rovnaký. Môžu sa vyskytovať kdekoľvek v reťazci. Formálne napísané: L(M,$\curvearrowright$) = $\{\text{w} \in \Sigma^* \mid \#_0(\text{w}) = \#_1(\text{w}) = \#_2(\text{w})\}$.

Príklad na GJFA

Máme daný GJFA M = (Q,Σ,R,s,F), kde:

- Q = $\{S_0, S_1\}$
- Σ = {a,b}
- R = $\{S_0\text{ba} \to S_1,\ S_1\text{a} \to S_1,\ S_1\text{b} \to S_1\}$
- s = S_0
- F = $\{S_0\}$

Tento automat akceptuje tie jazyky ktoré sú tvorené symbolmi a a b a zároveň sa tam nachádza podreťazec ba. Formálne napísané: L(M,$\curvearrowright$) = {a,b}* {ba} {a,b}*.

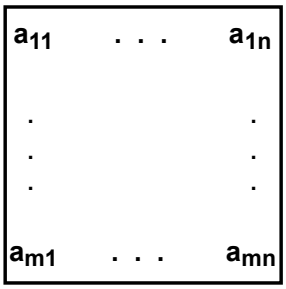
2.3 Dvojdimenzionálne automaty

Tieto druhy automatov majú na vstupe dvojrozmernú vstupnú pásku.

2.3.1 Základné pojmy

Vstup

Dvojrozmerný obrázok, alebo reťazec nad Σ je n-rozmerné obdĺžnikové pole, ktoré obsahuje prvky $a \in \Sigma$, pričom n má hodnotu dva. Všetky množiny takýchto reťazcov sú označované ako Σ**. Dvojrozmerný jazyk nad Σ je podmnožina Σ** [7].



Obr. 2.8: Vstupný obrázok p

Na obrázku 2.8 je znázornený vstupný obrázok p , ktorý má rozmery $m \times n$.

Veľkosť vstupu - obrázka

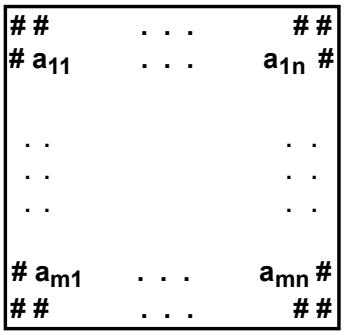
Nech existuje obrázok p ∈ Σ**, potom l_1 (p) označíme ako číslo riadkov a l_2 (p) označíme ako číslo stĺpcov obrázka p. Dvojica (l_1 (p), l_2 (p)) je označená ako veľkosť obrázka p. Prázdny obrázok má veľkosť (0,0) a je označený ako λ. Obrázky, ktoré majú veľkosť (0,n) alebo (n,0) kde n > 0 nie sú definované. Množina všetkých obrázkov nad Σ o veľkosti (m,n), kde m,n > 0 je označovaná ako $\Sigma^{m \times n}$. Avšak v prípade, že 1 < i < l_1(p) a 1 < i < l_2(p), potom označenie p(i,j) alebo p_{ij} označuje súradnice (i,j) obrázka p [7].

Podobrázok

Je to výrez, alebo určitá časť zo vstupu, kde musí platiť nasledovné. Máme daný obrázok p. Jeho rozmery sú (m,n). Podobrázok (alebo Blok) z obrázka p sa označuje ako p'. Veľkosť obrázka p' je (m',n'). Musí platiť, že $m' < m$ a súčasne $n' < n$, potom existujú čísla h a k tak, že ($h < m - m'$, $k < n - n'$) tak, že $p'(i,j) = p(i+h, j+k)$. Pre všetky i, j musí platiť $0 < i < m'$ a $0 < j < n'$ [7].

Hranice

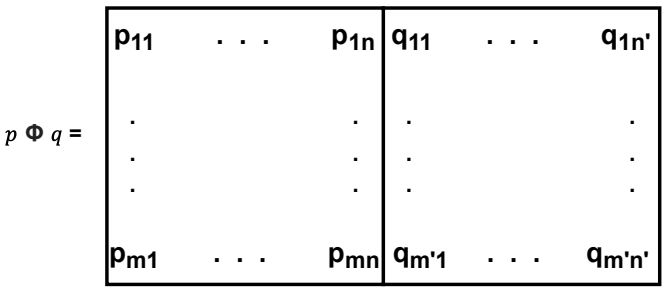
Hranice obrázka vieme identifikovať použitím špeciálneho symbolu #. Mám definovaný obrázok p o veľkosti (m, n). Pre každý p existuje obrázok s hranicami $\hat{p}$, ktorý bude mať veľkosť $(m+2, n+2)$. Ten je vytvorený obkolesením obrázka p špeciálnym hraničným symbolom #, ktorý nepatrí do Σ. $\# \notin \Sigma$. Znázornenie $\hat{p}$ je na obrázku 2.9 [7].



Obr. 2.9: Vstupný obrázok s hranicami $\hat{p}$

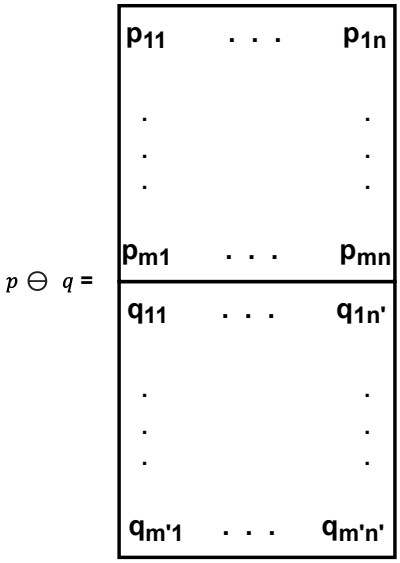
Konkatenácia obrázkov

Stĺpcovú konkatenáciu [7] (spojenie) dvoch obrázkov p a q označujeme ako $p \oplus q$. Operácia je definovaná iba v prípade, že $m = m$', čo znamená, že výšky oboch obrázkov sa musia rovnať. Veľkosť výsledného obrázka je ($n + n$',m).



Obr. 2.10: Stĺpcová konkatenácia p a q

Riadkovú konkatenáciu (spojenie) dvoch obrázkov p a q označujeme ako $p \ominus q$. Operácia je definovaná iba v prípade, že $n = n$', čo znamená, že šírky oboch obrázkov sa musia rovnať. Veľkosť výsledného obrázka je (n,$m + m$').



Obr. 2.11: Riadková konkatenácia p a q

2.3.2 Štvorcestný automat

V roku 1967 bol predstavený nový model konečného automatu, ktorý dokázal pracovať s dvojrozmernou páskou na vstupe. Predstavili a popísali ho M.Blum a C. Hewitt. Bola to jedna z prvých definícii na rozpoznávanie obrázkových jazykov. Táto verzia automatu je rozšírená verzia dvojcestného automatu, ktorý je opísaný v sekcii 2.1.4. Dokáže sa pohybovať po vstupnej páske v štyroch smeroch. Hore, dole, doľava a doprava.

Definícia

Nedeterministický [7] (alebo Deterministický) štvorcestný konečný automat (anglicky: four-way finite automaton) označovaný ako 4NFA (4DFA) je sedmica $(\Sigma, Q, \Delta, q_0, q_a, q_r, \delta)$ definovaná nasledovne:

- Σ je konečná abeceda vstupnej pásky
- Q je konečná množina všetkých stavov
- Δ je množina povolených smerov {L,R,U,D}
- q_0 je počiatočný stav, $q_0 \in$ Q
- q_a, q_r sú prijímajúce stavy (accepting) a odmietnuté stavy (rejecting), $q_a, q_r \in$ Q
- δ : Q $\setminus \{q_a, q_r\} \times \Sigma \to 2^{Q \times \Delta}$ ($\delta : Q \setminus \{q_a, q_r\} \times \Sigma \to Q \times \Delta$) je prechodová funkcia

Štvorcestný konečný automat sa najčastejšie označuje ako 4FA, kedy netreba špecifikovať, či ide o determinizmus, alebo nedeterminizmus. Na začiatku sa nachádza v počiatočnom stave q_0. Postupne na základe definovaných pravidiel prechádza po vstupnej páske rôznymi smermi d, $d \in \Delta$. Pohyb závisí na základe prechodovej funkcie δ, ktorá je daná stavom, v ktorom sa 4FA nachádza a symbolom na aktuálnej pozícii vstupu. Výsledkom je nový stav, do ktorého sa 4FA dostane a smer d, ktorým sa posunie. V momente, kedy sa aktuálny stav

nachádza v stave q_a, automat sa zastaví a vstup sa príjme. Pri rozpoznávaní to znamená, že vstup sa rozpoznal. Opačne, ak aktuálny stav nachádza v stave q_r, zastaví sa a vstup sa odmietne. Počítajme s tým, že každý vstup je obrázok s hranicami $\hat{p}$ 2.3.1. Ak sa pohybom dostaneme na pozíciu so symbolom #, v nasledujúcom ťahu sa vráti naspäť. Tým pádom by sa dalo povedať, že riadenie je citlivé na hranice. 4FA vie, kedy sa blíži ku hraniciam a nikdy tak neprekročí hranice obrázka, ktorý rozpoznáva.

Ako sme si mohli všimnúť, do akceptujúceho stavu sa môže dostať prakticky kedykoľvek, v závislosti od δ. Nie je nutné aby prešiel cez všetky pozície obrázka (prečítal celý obrázok). Môže sa ale vracať na pozície, v ktorých sa už nachádzal toľkokrát, koľko to je potrebné na akceptovanie (prijatie) vstupného obrázka.

Príklad 1

Nech je definovaný 4FA M, ktorý dokáže rozpoznať všetky jazyky L, pričom L $\subseteq \Sigma^{**}$. Abeceda $\Sigma = \{0,1\}$. Neformálne zapísané, do L patria všetky obrázky, ktoré majú na párnych riadkoch iba symbol 1 a na nepárnych riadkoch iba symbol 0.

Automat M sa pohybuje od ľavého horného symbolu smerom doprava. Pritom kontroluje aktuálny symbol. Ak naraz na hranice obrázka, posunie sa smerom nadol a pokračuje naspäť smerom doľava. Po detekcii hranice sa opäť posunie nadol a pohyb sa opakuje, kým neprejde celý vstup.

p =

```
# # # # # # # # # # # # # #
# 0 0 0 0 0 0 0 0 0 0 0 0 #
# 1 1 1 1 1 1 1 1 1 1 1 1 #
# 0 0 0 0 0 0 0 0 0 0 0 0 #
# 1 1 1 1 1 1 1 1 1 1 1 1 #
# # # # # # # # # # # # # #
```

Obr. 2.12: Vstupný obrázok $p \in L$

Na obrázku 2.12 je zobrazený obrázok p, ktorý je jeden z možných vstupov automatu M, kde platí, že $p \in L$.

Pravidlá δ takéhoto automatu M sú:

$q_0 0 \to q_1 R$
$q_1 0 \to q_1 R$
$q_1 \# \to q_2 D$
$q_2 \# \to q_3 L$
$q_3 \# \to q_a$
$q_3 1 \to q_4 L$
$q_4 1 \to q_4 L$
$q_4 \# \to q_5 D$
$q_5 \# \to q_6 R$
$q_6 \# \to q_a$
$q_6 0 \to q_1 R$

Príklad 2

Uvažujme jazyk L_1, $L_1 \subseteq \Sigma^{**}$, ktorého počet riadkov n a počet stĺpcov m je rovnaký a súčasne n a m sú nepárne čísla. V strede obrázka sa nachádza hodnota 1. Vstupná abeceda Σ je z množiny {0,1}. Príklad takéhoto jazyka je na obrázku 2.13.

Definujeme si 4NKA N, ktorý dokáže prijímať takéto jazyky L_1. Začína na pozícii (1,1). Pohybuje sa po diagonále smerom do stredu. V určitom bode sa uplatní nedeterminizmus. N si zapamätá aktuálny symbol na vstupe a začne sa pohybovať po ďalšej diagonále smerom dole. Ak N narazí na spodný ľavý roh, potom na vstupe je štvorec s nepárnou dĺžkou strany. Následne sa skontroluje zapamätaný symbol. Ak je totožný so symbolom 1, N končí úspešne, inač neúspešne.

p =

```
#######
#00000#
#00000#
#00100#
#00000#
#00000#
#######
```

Obr. 2.13: Vstupný obrázok $p \in L_1$

2.3.3 On-line Tesselačný automat

Doteraz boli opísané automaty, ktoré sa pohybovali sekvenčne po vstupnej páske. V každom kroku čítali jeden symbol. V tejto časti sa budeme zaoberať takzvanými bunkovými (anglicky: cellular) automatmi (CA). Skladajú sa z jednoduchých komponentov, ktoré voláme bunky. Väčšinou sú rovnaké. Každá bunka má svoj stav. V čase sa mení. Stav buniek CA je možné počítať v jednotlivých krokoch v diskrétnom čase pomocou pravidiel, ktoré počítajú nasledujúci stav zo stavu buniek a jeho okolia.

Existujú rôzne typy CA. Každý má špecifické pravidlá a rôzne definované susedské okolie. Jedny z najznámejších jednoduchých hier sú napríklad, hra Život (anglicky: Life), Langtonov mravec (anglicky: Langton's ant) a iné. Poprípade môžu byť použité na simuláciu chemických reakcií, dopravy až po modely umelého života.

On-line Tesselačné automaty sú konkrétne modely CA. Predstavené boli K. Inoue a A. Nakamura. Obsahujú pole buniek. Každá je v každom čase v nejakom stave. Bunky nevytvárajú v každom kroku prechody, ale v takzvaných prechodových vlnách (anglicky: transition wave). Pohybujú sa diagonálne naprieč celým poľom. V závislosti na dvoch susedoch (jeden zhora a ďalší zľava) každá bunka mení svoj stav.

Definícia cellulárych automatov

Celulárne automaty (anglicky: Celullar Automaton) je štvorica A = (d,S,N,f), zložená z nasledujúcich zložiek:

- Dimenzia d - Pole buniek je všeobecne n-rozmerné, obvykle 1D alebo 2D, môže byť konečné alebo nekonečné, pričom všetky bunky sú rovnaké.
- Konečná množina stavov S.
- Susedstvo N (anglicky: Neighbourhood) je počet a pozícia okolitých buniek, s ktorými bunka pracuje
- Pravidla f, ktoré popisujú chovanie buniek v čase.

Definícia On-line Teselačných automatov

Nedeterministický (Deterministický) dvojrozmerný on-line teselačný automat [7] označovaný ako 2OTA (2-DOTA) je kompletne definovaný ako pätica (Σ,Q,q_0,F,δ) kde:

- Σ je vstupná abeceda
- Q je konečná množina všetkých stavov
- I $\subseteq$ Q (I = {i} $\subseteq$ Q) je množina počiatočných stavov
- F $\subseteq$ Q, je množina konečných (povolených) stavov
- δ : Q $\times$ Q $\times$ $\Sigma \rightarrow 2^Q$ (δ : Q $\times$ Q $\times\Sigma \rightarrow$ Q) je prechodná funkcia

Rozpoznávanie pri 2OTA začína v počiatočnom stave q_0. Väčšinou je to na súradniciach (1,1). Každá bunka vstupného obrázka p na pozícii (i, j) obsahuje stav. Pozrie sa na svojich dvoch susedov na pozíciach $(i-1, j), (i, j-1)$ a aktuálny symbol. Pomocou prechodovej funkcie δ získa tak svoj stav. V ďalšom kroku získajú svoj stav bunky pod ňou.

Počet krokov potrebných na získanie stavu poslednej bunky je $l_r(p)+l_s(p)-1$. Označenie $l_r(p)$ je počet riadkov p a $l_s(p)$ je počet stĺpcov p.

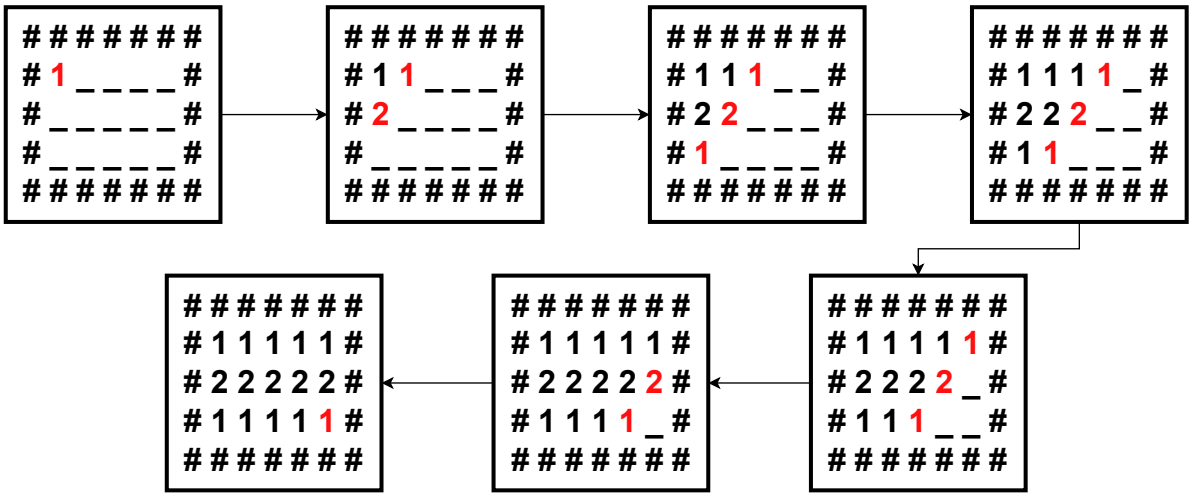
Príklad 1

Majme definovaný 2OTA M. Jeho formálna definícia je zapísaná nižšie. Pre abecedu Σ platí, Σ = {a}. Automat M prijíma jazyk L, pričom $L \subseteq \Sigma^{**}$. Jazyk L obsahuje všetky obrázky, ktoré majú nepárny počet riadkov.

2OTA M =

- Q = {0,1,2}
- I = {0}
- F = {1}
- δ(0,0,a) = δ(0,2,a) = δ(1,0,a) = δ(1,2,a) = 1
- δ(0,1,a) = δ(2,1,a) = 2

Znázornenie pohybu M v jednotlivých krokoch je na obrázku 2.14. Na vstupe je obdĺžnik p s rozmermi (5,3). V tomto prípade v bunke je napísaný stav, v ktorom sa nachádza, keďže symbol majú všetky rovnaký. Podľa predchádzajúceho vzorca $l_r(p) + l_s(p) - 1$ nám stačí na prechod celým obdĺžnikom iba 7 krokov. Po 7 kroku má už aj na poslednej pozícii (5,3) vypočítaný stav. Ten patrí do množiny koncových stavov, F a p je tak rozpoznané.



Obr. 2.14: Príklad pohybu 2OTA M

Kapitola 3

Dvojdimenzionálne skákajúce automaty

Sú spojením dvoch typov automatov, ktoré boli opísané v predchádzajúcej sekcii a to sú Skákajúce automaty 2.2 a Dvojdimenzionálne konečné automaty 2.3. Vzniká tak ďalší typ automatu, ktorý môže mať vplyv na niektoré časti informatiky, ako napríklad rozpoznávanie obrázkov. V tomto prípade skoky automatu umožňujú rýchlejší prechod vstupným obrázkom ako klasické automaty, ktoré majú na vstupne dvojrozmernú pásku. Takisto dokážu redukovať počet stavov.

3.1 Pojmy

3.1.1 Definícia

Dvojdimenzionálny skákajúci automat (anglicky: Two-Dimensional Jumping Automata (2DJA)) pozostáva z nasledujúcich častí:

- Konečná množina stavov, často označovaná ako Q
- Konečná množina vstupných symbolov, často označovaná ako Σ
- Množina povolených smerov Δ
- Množina pravidiel R_1 pre pohyb, ktorá je v tvare: $(Q \times (\Sigma^* \cup \boxtimes) \times \Delta) \to Q$
- Množina pravidiel R_2 pre skoky, ktorá je v tvare: $(Q \times (\Sigma^* \cup \boxtimes)) \to Q$
- Počiatočný stav s, ktorý je z množiny Q.
- Konečná množina koncových stavov, označovaná ako F. Pričom F $\subseteq$ Q.

Vstup

Vstupom do automatu 2DJA je dvojrozmerná páska, obrázok, poprípade hocijaký iný dvojrozmerný útvar. Pozostáva zo symbolov z množiny Σ. Po obvode vstupu sa nachádzajú hranice označené ako #, ktoré boli definované v sekcii 2.3.1.

Stavy

Množina stavov Q, obsahuje všetky stavy, do ktorých sa môže daný automat M dostať pri prechode vstupu. Počiatočný stav s je stav, z ktorého automat začína svoj pohyb. Automat končí úspešne, ak sa už nedokáže ďalej pohnúť a súčasne sa nachádza v jednom z konečných stavov F. Zastatie automatu môže byť spôsobene tým že prečítal celý vstup, alebo dostal sa do stavu z ktorého neexistuje prechod ďalej.

Množina povolených smerov Δ

Množina Δ obsahuje symboly {L,R,U,D}. Pričom musí platiť, že Σ - $\Delta = 0$. Tie určujú smer vo vstupnej páske, do ktorého sa pohne automat. Je to podobné ako pri Štvorcestných automatoch, ktoré sú opísané v sekcii 2.3.2. Povolené pohyby sú:

- L - (Left) pohyb doľava
- R - (Right) pohyb doprava
- U - (Up) pohyb nahor
- D - (Down) pohyb dole

Množiny $\mathbf{R_1}$ a $\mathbf{R_2}$

Sú to množiny pravidiel, na základe ktorých sa prechádza z jedného stavu do nasledujúceho. Najprv sa vyberajú pravidlá z množiny R_1, ktoré zabezpečujú pohyb. Ak neexistuje žiadne pravidlo R_1 potom sa vyberá pravidlo z množiny R_2 ktorá zabezpečuje skoky. Ak neexistuje žiadne pravidlo, ktoré by mohol automat aplikovať a tým sa pohnúť, tak sa skontroluje aktuálne v akom stave sa 2DJA nachádza. Keď patrí do množiny koncových stavov F, potom 2DJA skončil úspešne, ináč nie.

Vždy pri prechode cez bunku vstupnej pásky sa aktuálny symbol zmení na prečítaný symbol. Označuje sa ako $\boxtimes$, kde platí, že $\boxtimes \notin \Sigma$. Je to, preto aby automat nečítal dvakrát ten istý symbol.

Každá vstupná páska obsahuje hranice. Označujú sa symbolom #, ich definícia je v predchádzajúcej kapitole 2.3.1. Prechody za hranice obrázka nie sú povolené.

Pravidlá pre množinu R_1 sú v tvare $pad \to q$ pričom $p, q \in Q$, $a \in (\Sigma^* \cup \boxtimes)$ a $d \in \Delta$. Nech $x, a, z \in \Sigma^*$ je dvojrozmerná páska, pričom x sú predchádzajúce symboly, a je aktuálne čítaný symbol a z sú nasledujúce symboly. Pravidlá pre pohyby d = {R,D} modifikujú vstupnú pásku nasledovne: $xpaz \to x \boxtimes qz$. Pravidlá pre pohyby d = {L,U} modifikujú vstupnú pásku nasledovne: $xpaz \to xq \boxtimes z$.

Pravidlá pre množinu R_2 sú v tvare $pa \to q$, pričom $p, q \in Q$ a $a \in (\Sigma^* \cup \boxtimes)$. Nech $x, a, z \in \Sigma^*$ je vstupná dvojrozmerná páska, pričom x sú predchádzajúce symboly, a je aktuálne čítaný symbol a z sú nasledujúce symboly. Skok zapisujeme nasledovne $xpaz \curvearrowright xqz$.

3.1.2 Pohyb a skoky

Pohyb začína v počiatočnom stave s. Pomocou pravidiel z množiny $M = M_1 \cup M_2$ sa môže 2DJA pohybovať po páske a meniť tak svoj stav.

Skok v tomto type automatu značí presun čítacej hlavice z aktuálnej pozície na inú pozíciu. Preskočia sa tak niektoré bunky vstupnej pásky a prechod automatu je rýchlejší. Najčastejšie sa používajú, keď už na začiatku rozpoznávania máme informáciu o tom, kde približne by sa mala nachádzať konkrétna časť, ktorú hľadáme a pritom nás nezaujíma, čo sa nachádza inde.

Skoky na konkrétnu pozíciu vstupnej pásky

Nastáva v niektorých prípadoch, keď máme dostatok informácii o tom, kde by sa mala nachádzať hľadaná časť vo vstupe. Napríklad podobne ako je to v príklade 3.2. Čítacia hlavica tak môže skočiť na pozíciu, ktorá je definovaná riadkom a stĺpcom. Formálna definícia takéhoto skoku je:

$$xpaz \curvearrowright_{i,j} x'qz'$$

Značíme $p, q \in Q$, $x, x', a, z, z' \in (\Sigma^* \cup \boxtimes)$, kde x sú predchádzajúce symboly, a je aktuálne čítaný symbol a z sú nasledujúce symboly. Index i značí riadok a index j stĺpec, na ktorý sa presunie čítacia hlavica.

V prípade ak explicitne uvedieme znamienko pred indexom, značí to o koľko riadkov (stĺpcov) sa čítacia hlavica posunie od aktuálnej pozície. Kladné číslo značí posun dopredu a záporné posun dozadu.

Špeciálnym prípadom takejto operácie je skok na začiatok nasledujúceho riadku. V prípade, že sa posúva smerom zhora nadol, definícia na ďalší riadok môže vyzerať nasledovne:

$$xpaz \curvearrowright_{+1,1} x'qz'$$

Skoky na začiatok nasledujúceho riadku

Takýto typ skokov je najbežnejší. Ak automat prejde napríklad na koniec riadku, môže sa skočiť na začiatok nasledujúceho riadku. Klasický 2DKA automat by sa musel otočiť a prejsť na pozíciu nového riadku. Automatu 2DJA stačí vykonať skok a skoro v momente je na novej pozícii. Čítacia hlavica sa tak posunie na začiatok nového riadku. Formálna definícia takéhoto skoku je:

$$xpaz \swarrow x'\#qz'$$

Pričom $p, q \in Q$, $x, x', a, z, z' \in (\Sigma^* \cup \boxtimes)$, kde x sú predchádzajúce symboly, a je aktuálne čítaný symbol a z sú nasledujúce symboly.

3.2 Príklady

Príklad 1

Majme jazyk L, kde do L patria všetky obrázky p, v ktorých je obdĺžnik umiestnený tak, aby prechádzal stredom p. Čiže bod S = $(\frac{l_r(p)}{2}, \frac{l_s(p)}{2})$ sa musí nachádzať vo vnútri obdĺžnika. Hodnota $l_r(p)$ je počet riadkov p a hodnota $l_s(p)$ je počet stĺpcov p.

Máme automat 2DJA M. Vstupná abeceda $\alpha = \{0,1\}$, δ vyzerá nasledovne:

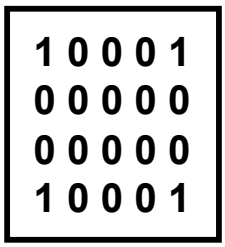
$$q_0 0 \curvearrowright_{Sx,Sy} q_1 \qquad q_2 0 \curvearrowright_{-1,-1} q_3 \qquad q_4 0 \curvearrowright_{+1,+1} q_5$$
$$q_1 0R \to q_1 \qquad q_3 1L \to q_3 \qquad q_5 1R \to q_5$$
$$q_1 1D \to q_2 D \qquad q_3 0 \curvearrowright_{-1,+1} q_4 \qquad q_5 0 \curvearrowright_{+1,+-} q_2$$
$$q_2 0D \to q_2 \qquad q_4 1U \to q_4 \qquad q_2 \boxtimes D \to q_a$$

M na začiatku skočí do stredu obrázka a pohybuje sa doprava, kým nenarazí na pravú stranu obdĺžnika. Postupne prechádza všetky hrany. Keď sa dostane na koniec jednej, tak skočí na začiatok ďalšej. Koniec rozpoznávania nastáva,keď sa dostane na pozíciu, v ktorej sme už boli (označená symbolom $\boxtimes$).

Príklad 2

Nech abeceda $\Sigma = \{0,1\}$. Označme si automat M taký, ktorý bude prijímať vstupy, kde v každom rohu sa bude vyskytovať symbol 1. R je počet riadkov a S je počet stĺpcov. Pravidlá pre takýto automat sú:

$$q_0 1 \curvearrowright_{1,S} q_1$$
$$q_1 1 \curvearrowright_{R,S} q_2$$
$$q_2 1 \curvearrowright_{R,1} q_a$$



Obr. 3.1: Príklad vstupu automatu M

Stav q_0 je počiatočný. Z neho sa skočí do pravého horného rohu, potom do pravého dolného a nakoniec do ľavého spodného rohu. Koncový stav je q_a.

3.3 Rozpoznávanie čísel

Nasledujúca časť ukazuje jeden z mnohých spôsobov, ako sa dá využiť 2DJA. V tomto prípade to je rozpoznávanie čísel zo vstupného obrázka.

Na vstupe sa hľadajú náznaky začiatku objektu, ktorý sa snažíme rozoznať. Naň sa aplikujú postupne rôzne typy automatov. To, aký automat sa použije, bude vyberať rozhodovací strom. Ak sa dostaneme na koniec stromu (do listovej časti), získame výsledok. Posúvame sa ďalej po obrázku a postup opakujeme, kým neprehľadáme celý vstup.

Tým dokážeme v jedom obrázku nájsť všetky čísla, ktoré sa v ňom nachádzajú, ak automat je schopný ich rozpoznať.

Pre zjednodušenie sa bude používať abeceda $\alpha = \{0,1\}$, kde 0 značí farbu povrchu (napríklad papier), na ktorom budú napísané znaky. Hodnota 1 naopak bude značiť farbu znaku (číslo). Predpokladá sa, že farba jedného čísla bude konštantná. Rovnako bude konštantná aj farba povrchu, poprípade sa nebude často meniť. Napríklad, keď sa odfotografuje papier a v pozadí vidno časť stola.

3.3.1 Jednoduché automaty

Používa sa viac automatov, ktoré dokážu rozoznať zopár jednoduchých útvarov. Každý dokáže rozpoznať čiary, ako vodorovná, zvislá alebo šikmá čiara. Líšia sa hlavne tým, ktorým smerom sa budú pohybovať. Pomenované sú podľa toho, z kade, kam idú.

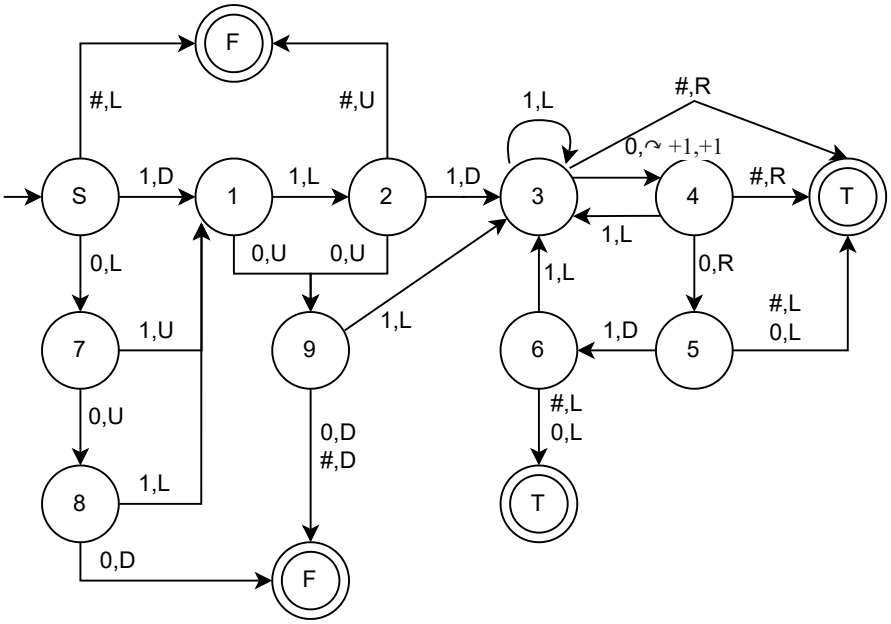
Každá hrana automatu, ktorá vedie od jedného stavu k druhému je označená dvoma symbolmi. Prvý je reprezentovaný množinou vstupných symbolov Σ a druhý je z množiny povolených smerov Δ.

Vo vybraných stavoch sa počíta. koľkokrát sa automat dostal do daného stavu. Pomáha to pri vyhodnotení, ktorý útvar sa našiel. Napríklad, ak sa automat nachádzal skoro po celú svoju dobu v stave, ktorým sa posúval smerom dole, tak je veľká šanca, že práve rozpoznal zvislú čiaru.

Up Left (UL)

Prvý typ automatu sa pohybuje po symboloch $1 \in \Sigma$ smerom doľava a snaží sa dostať do najľavejšieho spodného bodu. Počíta sa tu koľkokrát sa sa vošlo do stavu q_3 a q_4. Ak beh skončí v koncovom stave T, vyberie sa jeden z nasledujúcich výstupov.

- Zvislá čiara - Ak je počet vstupov do stavu q_3 a q_4 približne rovnaký.
- Vodorovná čiara - Ak je počet vstupov do stavu q_3 oveľa väčší ako q_4.
- Ináč nájdený útvar je krivá čiara, ktorá ide smerom zhora doľava.



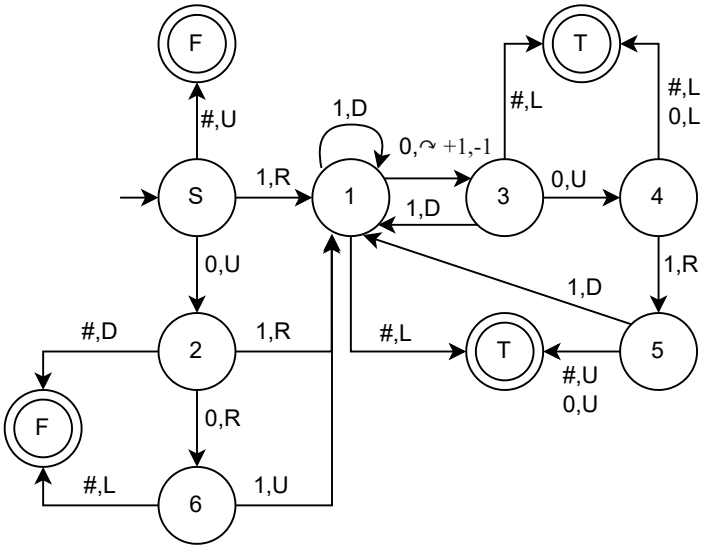
Obr. 3.2: Automat UL

Stavy q_1, q_2, q_7, q_8, q_9 kontrolujú či sa automat na začiatku môže pohybovať do ľavej strany. V stave q_3 sa vykonáva pohyb smerom doľava, až pokým neprečíta iný symbol. Pohne sa smerom dole a to sa opakuje kým sa už nemá kam pohnúť.

Left Down (LD)

Ďalší automat sa pohybuje po symboloch $1 \in \Sigma$ smerom dole a snaží sa dostať do najspodnejšieho pravého bodu. Počíta sa tu koľkokrát sa sa vošlo do stavu q_1 a q_3. Ak beh skončí v koncovom stave T, vyberie sa jeden z nasledujúcich výstupov.

- Zvislá čiara - Ak je počet vstupov do stavu q_1 oveľa väčší ako q_3.
- Vodorovná čiara - Ak je počet vstupov do stavu q_1 a q_3 približne rovnaký.
- Inač nájdený útvar je krivá čiara, ktorá ide smerom zhora doprava.



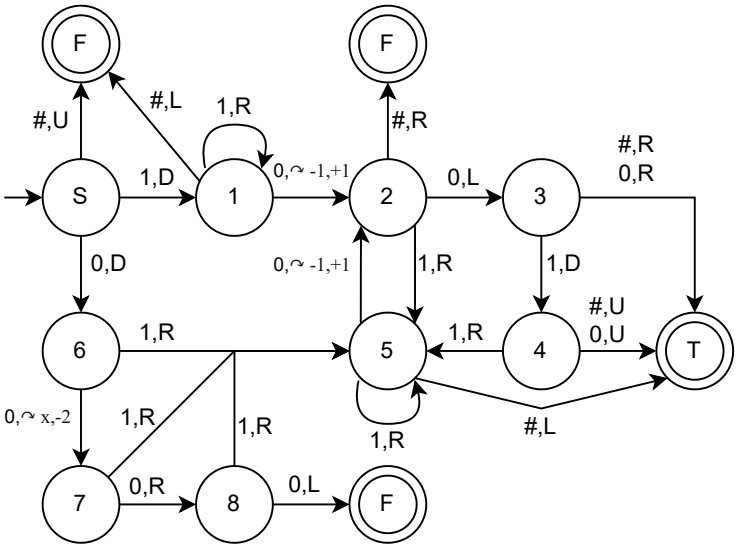
Obr. 3.3: Automat LD

V stave q_1 sa vykonáva pohyb smerom dole, až pokým neprečíta iný symbol. Pohne sa doprava a to sa opakuje, kým sa už nemá kam pohnúť.

Up Right (UR)

Nasledujúci automat sa pohybuje po symboloch $1 \in \Sigma$ smerom doprava a snaží sa dostať do najpravejšieho spodného bodu. Počíta sa tu koľkokrát sa sa vošlo do stavu q_2 a q_5. Ak beh skončí v koncovom stave T, vyberie sa jeden z nasledujúcich výstupov.

- Zvislá čiara - Ak je počet vstupov do stavu q_2 a q_5 približne rovnaký.
- Vodorovná čiara - Ak je počet vstupov do stavu q_5 oveľa väčší ako q_2.
-Ináč nájdený útvar je krivá čiara, ktorá ide smerom zhora doprava.



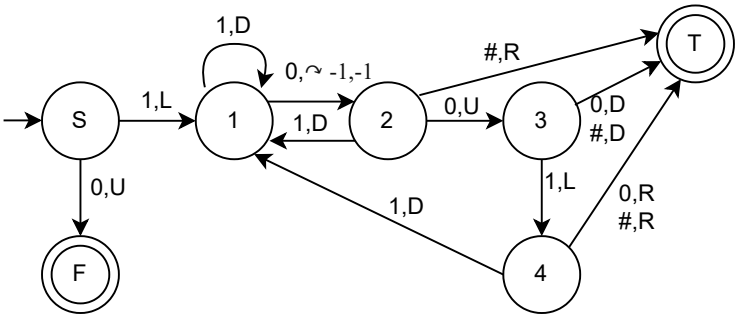
Obr. 3.4: Automat UR

Stavy q_6, q_7, q_8 kontrolujú či sa automat na začiatku môže pohybovať do pravej strany. V stave q_1 sa čítacia hlavica presúva čo najviac do pravej strany, z kade sa následne bude rozoznávať útvar. V stave q_5 sa vykonáva pohyb smerom doprava, až pokým neprečíta iný symbol. Pohne sa dole a to sa opakuje, kým sa už nemá kam pohnúť.

Right Down (RD)

Tento typ automatu sa pohybuje po symboloch $1 \in \Sigma$ smerom dole a snaží sa dostať do najľavejšieho spodného bodu. Počíta sa tu koľkokrát sa sa vošlo do stavu q_1 a q_2.Ak beh skončí v koncovom stave T, vyberie sa jeden z nasledujúcich výstupov.

- Zvislá čiara - Ak je počet vstupov do stavu q_1 oveľa väčší ako q_2.
- Vodorovná čiara - Ak je počet vstupov do stavu q_1 a q_2 približne rovnaký.
- Ináč nájdený útvar je krivá čiara, ktorá ide smerom zhora doľava.



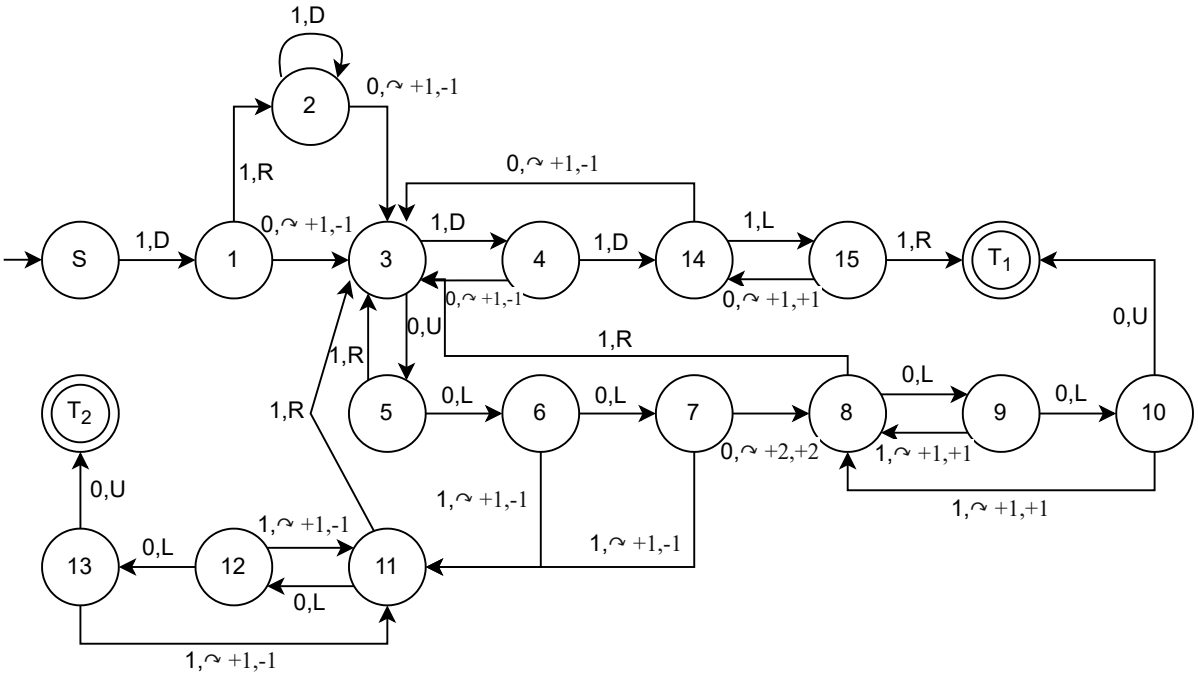
Obr. 3.5: Automat RD

V stave q_1 sa vykonáva pohyb smerom dole, až pokým neprečíta iný symbol. Pohne sa doľava a to sa opakuje kým sa už nemá kam pohnúť.

Automat Right

Líši sa od predchádzajúcich automatov tým, že nemá určený presný smer, ktorým sa bude pohybovať, ale opisuje hranicu objektu. Primárny smer je vpravo. Môže ísť hore aj dole. Koniec nastáva v prípade, že by sa musel pohnúť smerom naspäť, čiže doľava.

V prípade, že automat zastaví v stave T_1 výstupom je šikmá čiara, ak zastaví v stave T_2 tak je výstupom horizontálna čiara.

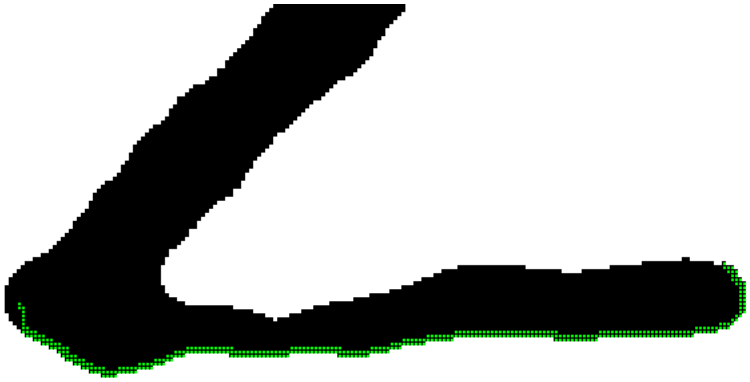


Obr. 3.6: Automat Right

Na obrázku 3.6 je zobrazený daný automat, stav S je počiatočný stav. Prvé dva stavy q_1 a q_2 zaisťujú, aby sa automat dostal na spodný okraj objektu. Odtiaľ sa pohybuje smerom doprava, zaisťuje to slučka medzi stavmi q_3 a q_4.

Ak sa automat nachádza v stave q_3 a na vstupne nieje znak "1", tak sa čítacia hlavica posúva smerom nahor (stavy q_5, q_6, q_7). Hľadá sa symbol "1", aby mohlo pokračovať s pohybom doprava. Ak je jediný možný pohyb dozadu, ide sa do stavu q_{12}, čo vedie ku rozpoznaniu horizontálnej čiary.

Ak sa pri pohybe nahor nenájde symbol "1", skúsi sa cesta dole (stavy q_8, q_9). Ak je tu jediný možný pohyb dozadu, ide sa do stavu q_{10}, čo vedie ku rozpoznaniu šikmej čiary.



Obr. 3.7: Príklad čísla 2

Obrázok 3.7 zobrazuje pohyb automatu *LineRight* na číslici 2, ktorý práve rozoznal horizontálnu čiaru.

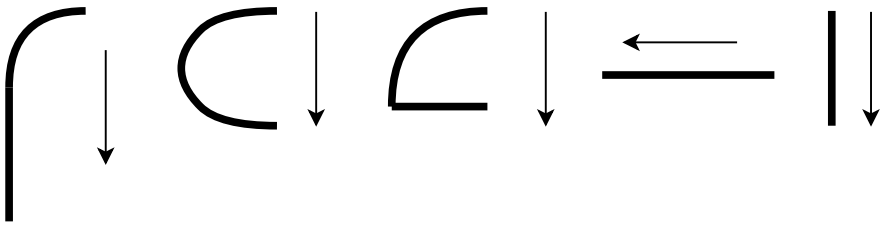
3.3.2 Spojené automaty

Pri aplikovaní jednoduchých automatov z predchádzajúcej kapitoly sa často stávalo, že po spustení jedného automatu sa bezprostredne po ňom spustil ďalší. Vznikali tak dvojice automatov. Ak sa spustil jeden z dvojice automatov, vedelo sa, že po ňom treba spustiť ďalší, vždy ten istý.

Táto kapitola je o spájaní jednoduchých automatov, ktoré po sebe vždy bezprostredne nasledujú. Tým pádom sa tak sprehľadnil a zjednodušil ich zápis. Vzniká tu zároveň abstrakcia, kedy sa nemusíme zaoberať, ako sa bude rozpoznávať jednotlivý útvar, ale z akých jednoduchých útvarov sa vytvorí zložitejší.

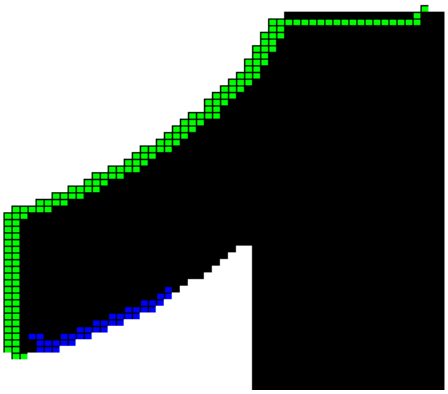
Automat A

Tento automat spája dva jednoduché automaty a to UL (3.3.1) a LD (3.3.1). Čítacia hlavica sa tak pohybuje po ľavej strane rozpoznávaného objektu. Najprv sa dostane do najľavejšej časti a potom do najspodnejšej. Jednotlivé útvary, ktoré sa rozpoznávajú sú nasledovné:



Obr. 3.8: útvary A

Šípky pri každom útvare znázorňuje smer, akým sa postupuje rozpoznávanie. Teoreticky výstup automatu môže byť kombinácia všetkých útvarov z výstupov UL a LD. Vybraté sú ale tie, ktoré sa používajú v následnej implementácii.

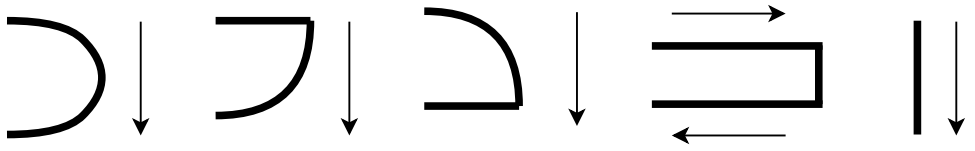


Na obrázku 3.9 je znázornený priebeh automatu na čísle 1. Zelené pixely reprezentujú pohyb automatu UL a modré pixely priebeh automatu LD. Oba sa snažia kopírovať obrysy objektu.

Obr. 3.9: Príklad A

Automat B

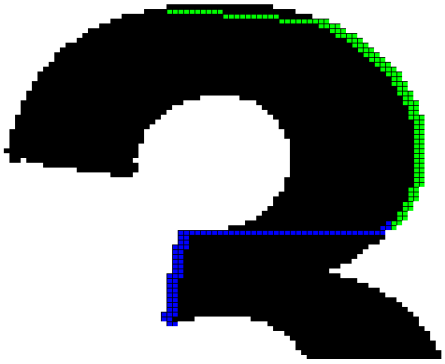
Je zložený z dvoch jednoduchých automatov a to UR (3.3.1) a UL (3.3.1). Čítacia hlavica sa pohybuje po pravej strane rozpoznávaného objektu. Najprv sa dostane do najpravejšej časti a potom do najspodnejšej. Jednotlivé útvary, čo sa dajú rozpoznať sú nasledovné.



Obr. 3.10: útvary B

Šípky pri každom útvare znázorňuje smer, akým sa postupuje rozpoznávanie. Teoreticky výstup automatu môže byť kombinácia všetkých útvarov z výstupov UR a UL. Vybraté sú ale tie, ktoré sa používajú v následnej implementácii.

Na obrázku 3.11 je znázornený ako postupuje automat pri rozpoznávaní na čísle 3. Zelené pixely reprezentujú pohyb automatu UR a modré pixely priebeh automatu UL. Oba sa snažia kopírovať obrysy objektu.



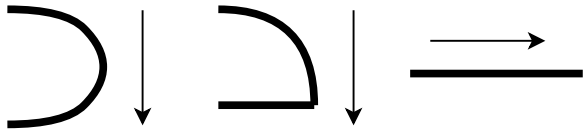
Obr. 3.11: Príklad B

Automat C

Podobne ako predchádzajúce, aj tento je zložený z dvoch automatov *Automat Right* a *RD*. Čítacia hlavica sa tu pohybuje smerom doprava. Následne sa pohybuje smerom hore alebo dole podľa situácie, v ktorej sa nachádza.

V prípade že výsledkom automatu *Right* je vodorovná čiara, automat C končí. Inač sa pokračuje ďalej.

Výsledky sa tak redukujú na nasledujúce možnosti:



Obr. 3.12: útvary C

Šípky pri každom útvare znázorňuje smer, akým sa postupuje rozpoznávanie. Teoreticky výstup automatu môže byť kombinácia všetkých útvarov z výstupov *Right* a *RD*. Vybraté sú ale tie, ktoré sa používajú v následnej implementácii.

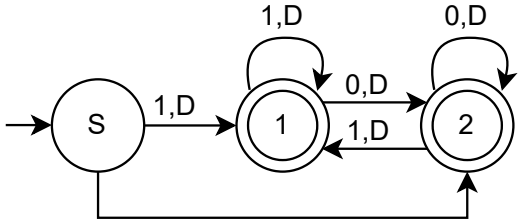
3.3.3 Špeciálne automaty

Predchádzajúce automaty, rozpoznávali jednoduché objekty, rôzne druhy čiar. Nasledujúce automaty sú zamerané hlavne na špeciálne operácie. Väčšinou sa pohybujú iba jedným smerom.

Line Down

Má dva body, počiatočný a koncový bod. Čítacia hlavica sa tu pohybuje od počiatočného bodu, až pokým sa nedostane na úroveň koncového bodu. Po ceste sa počíta, koľkokrát sa zmenil symbol (zmena symbolov z "1" $\rightarrow$ "0" a "0 $\rightarrow$ "1"). Detekuje počet hrán objektu, cez ktoré sa dostal pri svojom pohybe.

Využíva sa na presnejšie učenie výsledného čísla v prípade, že sa rozhodovací strom rozhoduje medzi číslami {0,8,9}.

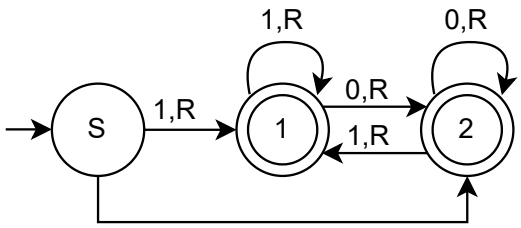


Stav S je počiatočný. V stave q_1 sa nachádza automat, ak prechádza cez symboly "1", ináč je v stave q_2. Počítadlo hrán sa inkrementuje pri prechode zo stavu q_1 do q_2, alebo opačne, z q_2 do q_1.

Obr. 3.13: Automat Line Down

Line Right

Používa sa len pri číslici 4. Je takmer totožný ako predchádzajúci automat *LineDown*. Rozdiel je ale v tom, do ktorej strany sa posúva čítacia hlavica. V prípade *LineDown* to bolo smerom dole, tu je to smerom doprava. Tiež sa tu počíta, koľkokrát sa detekovala hrana (zmena symbolov z "1" $\rightarrow$ "0" a "0 $\rightarrow$ "1").

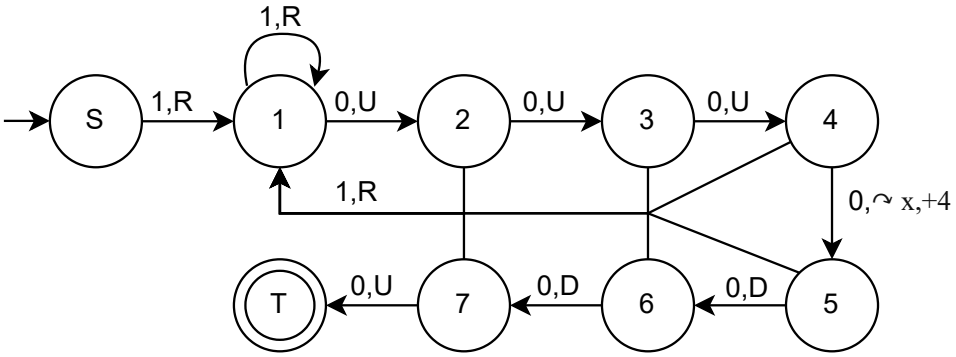


Stav S je počiatočný. V stave q_1 sa nachádza automat, ak prechádza cez symboly "1", ináč je v stave q_2. Počítadlo hrán sa inkrementuje pri prechode zo stavu q_1 do q_2, alebo opačne, z q_2 do q_1.

Obr. 3.14: Automat Line Right

Špeciálny S

Je využívaný len pri rozhodovaní, či ide o číslo 1, alebo 4. Jeho pohyb ide smerom zľava doprava. Počíta sa tu koľko symbolov prečítal. Ak je tento počet malý, pravdepodobne bude rozpoznaná číslica 1, ináč 4.



Obr. 3.15: Automat S

Ako je znázornené na obrázku 3.15, v stave q_1 sa pohybuje čítacia hlavica smerom doprava. V prípade že narazí na symbol "0", skúša sa pozerať až do 3 buniek na symboly nad a pod aktuálnym symbolom. Nájde symbol "1", pokračuje smerom doprava, ak nie, končí.

Kapitola 4

Stromové dátové štruktúry

Na začiatku tejto kapitoly sú opísané stručne základné pojmy o stromových dátový štruktúrach. Pomocou takýchto dátových štruktúr je riadené rozpoznávanie číslic z obrázka.

4.1 Základné pojmy

V informatike je strom veľmi všeobecná a výkonná dátová štruktúra, ktorá sa podobá skutočným stromom. Pozostáva z usporiadanej množiny prepojených uzlov v prepojenom grafe, v ktorom každý uzol má najviac jeden nadradený (rodičovský) uzol a nula alebo viac podriadených (potomkovských) uzlov v špecifickom poradí.

4.1.1 Všeobecná špecifikácia

Pozostáva z uzlov a hrán. Strom môže byť definovaný nasledovne:

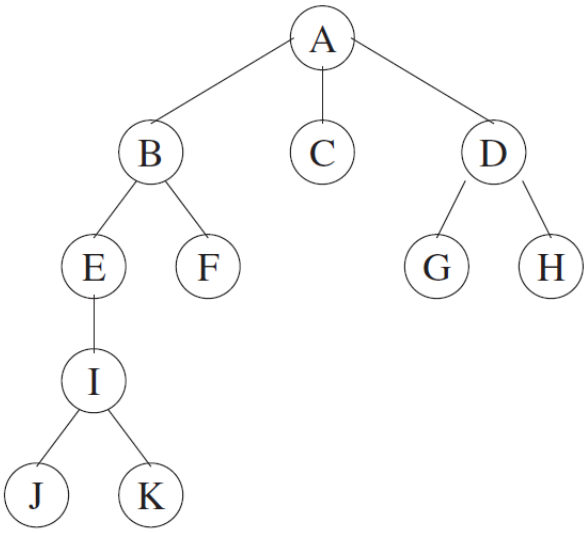
- Strom bez koreňa (anglicky: unrooted tree) - je definovaný ako graf (množina vrcholov a množina hrán, ktoré spájajú dvojice vrcholov) tak, že medzi akýmikoľvek dvoma vrcholmi v grafe existuje jedinečná cesta. Minimálna kostra grafu je príklad stromu bez koreňa.
- Koreňový strom (anglicky: rooted tree) - je konečná množina jedného alebo viacerých uzlov takých, že:
  - Existuje špeciálny uzol nazývaný koreň
  - Ostané uzly sú rozdelené do n disjunktných množín $T_1..T_n$, kde každá z týchto množín je stromom. $T_1..T_n$ sú nazývané podstromy koreňa.
- K-árny strom (anglicky: k-ary tree) je konečná množina uzlov, ktorá je buď prázdna, alebo pozostáva z koreňa a k prvkov disjunktných k-árnych stromov. Binárny strom je špeciálny prípad, kedy sa $k = 2$. [6]

Cesta

Cesta je postupnosť spojených hrán z jedného uzla do druhého. Stromy majú vlastnosť, že pre každý uzol existuje jedinečná cesta, ktorá ho spája s koreňom. [6]

Výška stromu

Maximálna dĺžka cesty v strome sa tiež nazýva výška stromu. Cesta maximálnej dĺžky vždy vedie od koreňa k listu. Veľkosť stromu je daná počtom uzlov, ktoré obsahuje. Strom pozostávajúci iba z jedného uzla má výšku 0 a veľkosť 1. Prázdny strom má veľkosť 0 a jeho výška definovaná ako -1. [6]



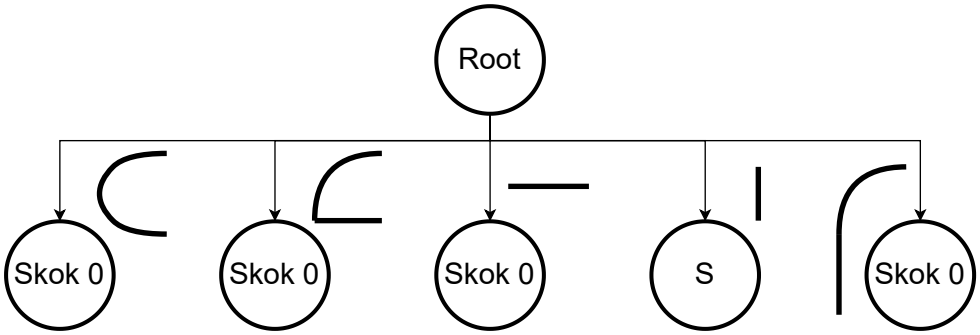
Obr. 4.1: Príklad stromu

Na obrázku 4.1 je znázornený strom. Uzly sú zvyčajne, aj keď nie vždy, označené dátovou položkou (napríklad číslom alebo kľúčom vyhľadávania). Označenie uzla sa bude označovať jeho hodnotou. Uzol A je koreňový uzol a má potomkov uzly B, C a D.

4.2 Rozpoznávanie riadené stromom

Pomáha nám určiť, ako bude prebiehať rozpoznávanie. Každý uzol má niekoľko nasledovníkov. Obsahuje informáciu o tom, ktorý automat sa spustí, kde sa presunie čítacia hlavica, poprípade, či sa spustí podmienka, ktorá porovná určité hodnoty.

Rozhodovanie sa začína v uzle označenom ako *Root*. Spustí sa spojený automat A (3.3.2). Tým pádom sa kontroluje ľavá časť rozpoznávaného útvaru. Na základe výsledkov sa prejde do ďalšieho uzla. Názov automatu v uzle znamená, ktorý automat sa prevedie. Symboly na prechodoch medzi uzlami označujú na základe ktorého výstupu sa prejde do nasledujúceho uzla.



Obr. 4.2: Začiatok stromu

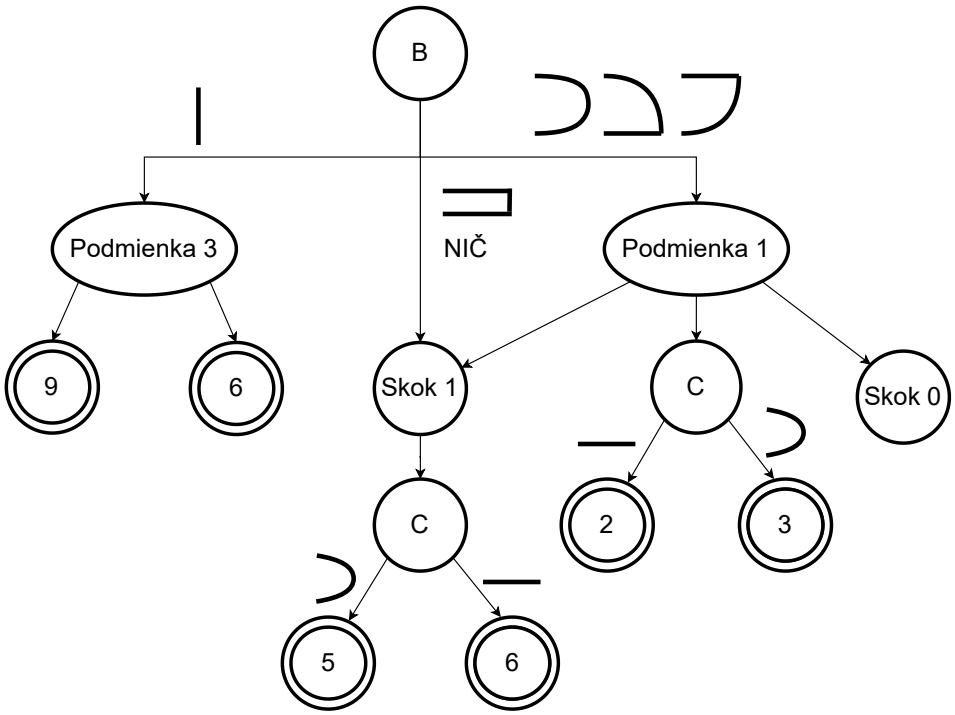
Na obrázku 4.2 je zobrazená časť stromu, každá vetva je opísaná v samostatnej sekcii. V týchto prípadoch skok znamená, že sa čítacia hlavica presunie na rovnaké miesto ako sa začínalo v uzle Root.

Základné typy uzlov

- **Automat** - vykoná sa jeden z nasledujúcich automatov {A,B,C Line Down, Line Right, alebo S}
- **Skok** - skočí sa na konkrétnu pozíciu
  - Skok 0 - pozícia začiatku rozpoznávania
  - Skok 1 - pozícia, na ktorej skončila prvá časť spojeného *Automatu A* (čiže automat UL)
  - Skok 2 - pozícia kde skončil spojený *Automat A*
  - Skok 3 - pozícia, na ktorej skončila prvá časť spojeného *Automatu B* (čiže automat UR)
  - Skok 4 - pozícia, kde skončil spojený *Automat B*
- **Podmienka** - porovnajú sa určité hodnoty, bližšie sú opísané pri jednotlivých číslach nižšie
- **Listový uzol** - je znázornený dvoma kružnicami, končí sa tu rozpoznávanie a výsledkom je rozpoznané číslo

4.2.1 Vetva 1

Ak sme sa dostali do prvého uzla, množina, čísel, ktoré môžu byť rozpoznané sa zúžila na 0,2,3,5,6,8,9. Opäť sa čítacia hlavica nachádza na rovnakej pozícii, ako na začiatku rozpoznávania. Tentoraz sa prehľadáva pravá časť čísla. Spustí sa spojený *Automat B*. Na základe jeho výsledku sa prejde do ďalšieho uzla.



Obr. 4.3: Prvá vetva stromu

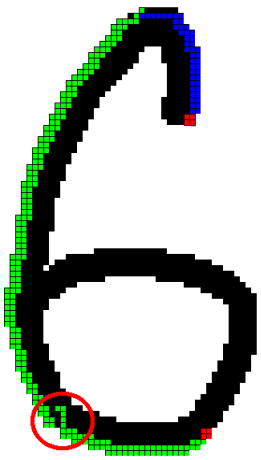
Čísla 6,9

Jedným s výsledkov môže byť vertikálna čiara. Vtedy sa rozhodovanie zúži iba na čísla 6 a 9. Jeden z príkladov, ako môže vyzerať v takomto prípade číslica 6, je ukázaný na obrázku 4.4. Aby sa dali tieto dve čísla rozlíšiť, vykoná sa podmienka s označením 3.

Podmienka 3

Aktuálnu hodnotu čítacej hlavice si označme ako bod B. Bod A bude pozícia čítacej hlavice pred prevedením *Automatu A*

- Výška bodu B (súradnica Y) je väčšia ako výška bodu A → rozpoznala sa 9.
- Výška bodu B (súradnica Y) je menšia ako výška bodu A → rozpoznala sa 6.



Obr. 4.4: Ukážka čísla 6

Na obrázku 4.4 zelená farba znázorňuje pohyb automatu A. Miesto, ktoré je zakrúžkované, ukazuje na časť, kedy sa končí jeho jedna časť a začína druhá. Je to zároveň aj miesto kam sa presunie čítacia hlavica pri uzle s názvovom Skok 1.

Modrá farba zobrazuje pohyb automatu B a jeho výstupom je akurát vertikálna čiara smerom nadol. Červené značky označujú body A (ten vyššie) a B (ten nižšie).

Čísla 5,6

Ak sa nenájde nič, alebo je výsledkom 2 krát horizontálna čiara, je veľká šanca, že sa našlo číslo 5 alebo 6. Nachádzame sa tak v uzle s hodnotou *Skok 1.* Čítacia hlavica sa presunie (prevedie skok).

Kebyže útvar, ktorý rozpoznávame je číslo 6, bod A by sa tak nachádzal na spodnej časti čísla. Znázornený je na obrázku 4.4 ako spodný červený bod.

V prípade, že je to číslo 5, nachádzal by sa približne v strede čísla. Názorná ukážka tohto bodu je na obrázku 4.7 označená červeným symbolom. Preto sa spustí ešte *Automat C* a jeho výsledok rozhodne o konečnom výsledku.

Podmienka 1

Doteraz boli opísané prvé dve vetvy, z ktorých sa dalo dostať buď do uzla z názvom Podmienka 3 alebo Skok 1.

Aktuálnu hodnotu čítacej hlavice si označme ako bod B. Bod A bude pozícia čítacej hlavice pred prevedením *Automatu A*. Bod R bude počiatočný bod, kde sa začínalo rozpoznávanie. Pre podmienku s označením 1 platí, nasledovné:

- Výška bodu B (súradnica Y) je väčšia ako výška bodu A → zúženie na čísla 2,3. Nasleduje uzol s hodnotou C
- Vzdialenosť |AR| medzi bodmi A a R je oveľa väčšia ako vzdialenosť |BR| medzi bodmi B a R → zúženie na čísla 5,6. Nasleduje uzol s hodnotou Skok 1
- Ak neplatia prvé dva body → zúženie na čísla 0,8,9. Nasleduje uzol s hodnotou Skok 0

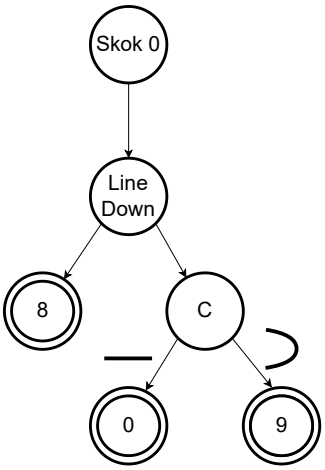
Čísla 2,3

Čítacia hlavica sa nachádza na spodnej strane rozpoznávaného objektu. Aby sa dalo rozoznať, či ide o číslo 2 alebo 3, spustí sa spojený automat C. Pre výsledok horizontálna čiara je to 2. Ak by to mala byť trojka, musela by sa nájsť ešte šikmá čiara, čo predstavuje spodnú časť čísla.

Čísla 0,8,9

Po vykonaní skoku na začiatok sa spúšťa *Automat LineDown*. Pohybuje sa, až kým nedôjde do najspodnejšieho bodu (bod z najväčšou súradnicou Y), cez ktorý sa prešlo pri pohybe. Výsledkom tohto automatu je číslo, ktoré reprezentuje koľkokrát sa našla hrana. Ak je počet hrán väčší ako 3, je veľká pravdepodobnosť, že sa prešlo cez úsek, kde sa križujú čiary v čísle 8. Rozpoznalo sa tak číslo 8. Ak nie, môže to byť ešte číslo 0, alebo 9. Aby sa dali rozlíšiť, spustí sa automat C.

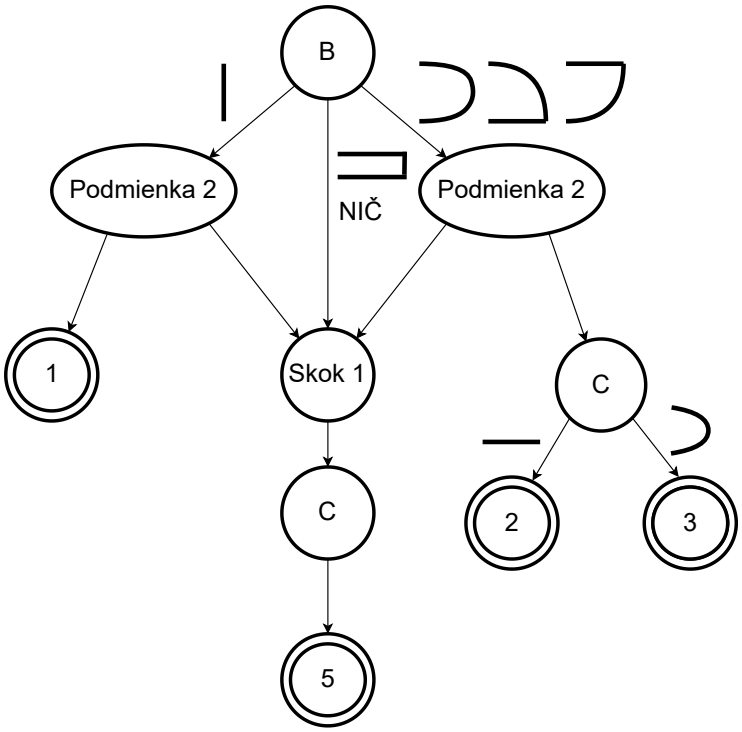
Na obrázku 4.5 je zobrazená časť stromu, ktorá pokračuje z predchádzajúcej podmienky 1.



Obr. 4.5: Strom pre čísla 0,8,9

4.2.2 Vetva 2

Ak sme sa dostali do druhej vetvy, množina čísel, ktoré môžu byť rozpoznané, sa zúžila na 1,2,3 a 5. Výsledok predchádzajúceho *Automatu A* musel tak byť šikmá čiara a horizontálna čiara. V tomto uzly už nie je toľko možností. Opäť sa čítacia hlavica nachádza na rovnakej pozícii, ako na začiatku rozpoznávania. Tentoraz sa ide prehľadávať pravá časť čísla. Spustí sa spojený *Automat B*. Na základe výstupu sa prejde do ďalšieho uzla.

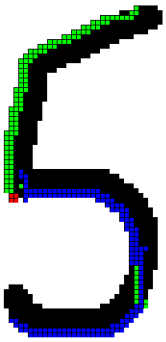


Obr. 4.6: Druhý vetva stromu

Na obrázku 4.6 je znázornená časť rozhodovacieho stromu. Listové uzly obsahujú čísla 1,2,3 alebo 5. Na základe podmienky. ktorá je opísaná nižšie, sa vyberá číslo medzi 1 a 5, alebo 5 a 2,3.

Číslo 5

Po rozpoznaní útvaru, ktorý pozostáva z dvoch horizontálnych čiar, alebo ak sa nerozpozná nič, tak sa výstup zúži iba na číslicu 5. Rozpoznávanie sa teraz nachádza v uzle označenom Skok 1. Čítacia hlavica sa tak presunie do určenej pozície (miesto kde skončia prvá časť automatu A), označenom na obrázku 4.7 červenou farbou. Odtiaľ sa spustí *Automat C*. Kontroluje sa spodná časť čísla 5. V prípade, že je výsledok útvar tvorený dvoma šikmými čiarami, rozpoznalo sa číslo.



Obr. 4.7: Ukážka čísla 5

Na obrázku 4.7 je zobrazená číslica 5. Zelené symboly predstavujú pohyb automatu A. Červený symbol prestavuje miesto, kde skončila jeho prvá časť. Druhá časť pokračuje smerom dole. Väčšina symbolov sa prekrýva s modrými symboly, ktoré predstavujú pohyb automatu C.

Podmienka 2

Aktuálnu hodnotu čítacej hlavice si označme ako bod B. Bod A bude pozícia čítacej hlavice pred prevedením *Automatu A*. Pre podmienku s označením 2 platí nasledovné:

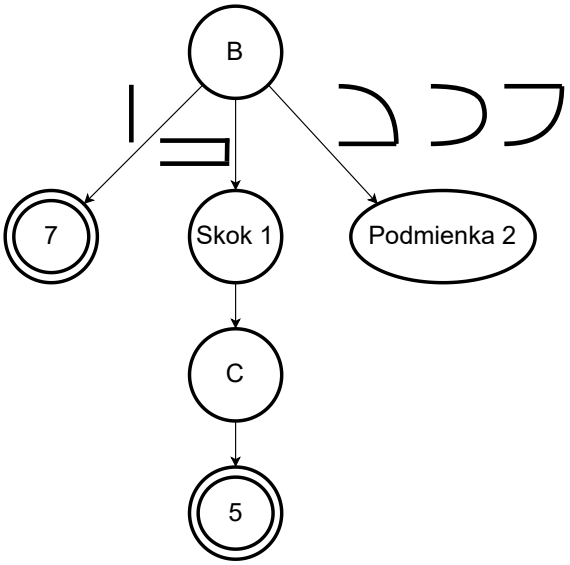
- Výška bodu B (súradnica Y) je väčšia ako výška bodu A → rozpoznala sa 1, poprípade je možné že sú tu čísla 2 alebo 3. zavisí to, z ktorej vetvy stromu sa zavolala daná podmienka. Znázornené je to na obrázku 4.6, kde sa nachádza časť rozhodovacieho stromu.
- Výška bodu B (súradnica Y) je menšia ako výška bodu A → rozpoznala sa pravdepodobne 5. Musí sa to ešte ďalej overiť.

4.2.3 Vetva 3

Do tohto uzla sa dostaneme v prípade, že prvý automat našiel horizontálnu čiaru. Výber sa tak zúžil medzi čísla 2,3,5 a 7. Opäť sa spúšťa *Automat B*, ktorý prechádza po pravej strane rozpoznávaného objektu.

V prípade, že sa nájde vertikálna čiara, rozpoznala sa číslica 7. Ak je výsledok dvakrát horizontálna čiara, výber sa zúžil na číslo 5. Rovnako, ako bolo opísané v predchádzajúcej sekcii, sa spustí ďalší automat a pri správnom výstupe sa nájde číslica 5.

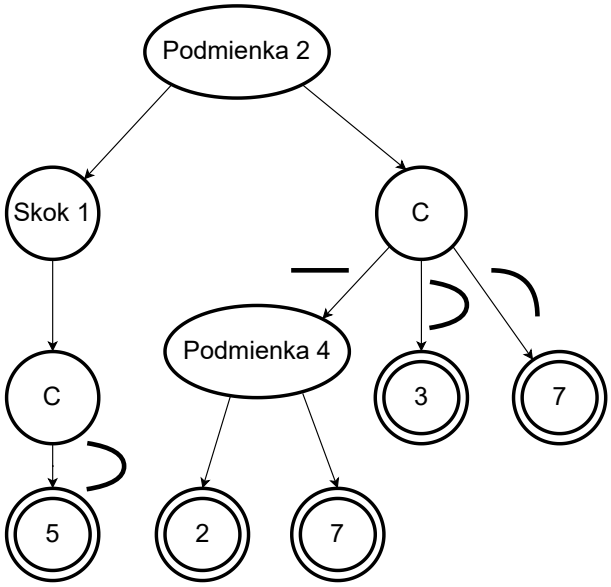
Je tu ešte podmienka s označením 2, do ktorej sa dá dostať pomocou troch útvarov, ktorá rozhoduje medzi číslami 2,3 a 5.



Obr. 4.8: Tretia vetva stromu

Aktuálnu hodnotu čítacej hlavice si označme ako bod B. Bod A bude pozícia čítacej hlavice pred prevedením *Automatu A*. Pre podmienku s označením 2 platí, nasledovné:

- Výška bodu B (súradnica Y) je väčšia ako výška bodu A → pravdepodobne sa rozpoznali čísla 2,3 alebo 7. Znázornené je to na obrázku 4.9, kde sa nachádza časť rozhodovacieho stromu.
- Výška bodu B (súradnica Y) je menšia ako výška bodu A → rozpoznala sa pravdepodobne 5. Musí sa to ešte ďalej overiť. Overuje sa tým, že sa spustí automat C.



Obr. 4.9: Strom pre čísla 2,3,5 a 7

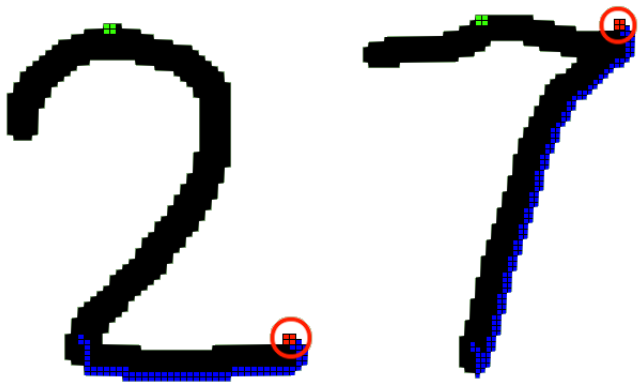
Po podmienke 2 sa môže spustiť *Automat C*. V prípade, že rozpoznal horizontálnu čiaru, na výber je ešte z troch čísel 2, alebo 7. Spustí sa tak ešte podmienka (uzol s hodnotou Podmienka 4), ktorá rozhodne o výsledku.

Ak sa rozpozná útvar iba z jednou šikmou čiarou, znamená to, že čiara po ktorej sa automat pohyboval, končí dole a nepokračuje smerom doľava, ako je to pri čísle 3. Našlo sa číslo 7. Ak útvar obsahuje dve takého šikmé čiary, dá sa povedať, že sa našlo číslo 3.

Podmienka 4

Aktuálnu hodnotu čítacej hlavice si označme ako bod C. Bod R bude počiatočná pozícia čítacej hlavice. Pre podmienku s označením 4 platí, nasledovné:

- Výška bodu R (súradnica Y) je väčšia alebo približne rovnaká ako výška bodu C → rozpoznalo sa číslo 7
- Výška bodu R (súradnica Y) je menšia ako výška bodu C → rozpoznalo sa číslo 2



Obr. 4.10: Ukážka podmienky 4

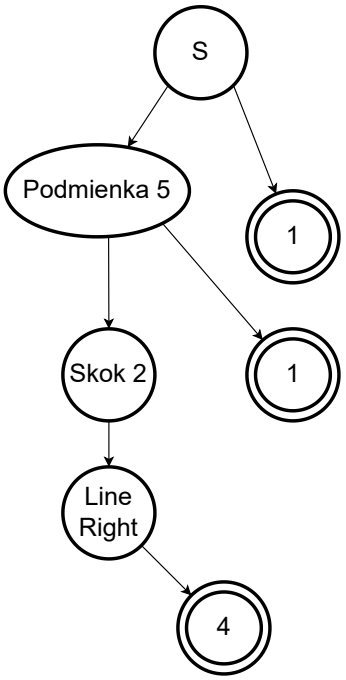
Na obrázku 4.10 je zobrazené ako funguje podmienka, ktorá rozhoduje či sa rozpoznalo číslo 2, alebo 4. Zelené symboli znázorňujú počiatok rozpoznávania (bod R). Modré symboly pohyb automatu C a červený symbol zas miesto, kde skončil (bod C).

4.2.4 Vetva 4 a 5

Nachádzame sa v predposlednej 4 vetve uzla Root. Výsledkom predchádzajúceho automatu bola čiara dole. Výber sa zúžil už iba na čísla 1,4.

V tomto prípade sa počíta s tým, že jednotka vyzerá ako čiara, ktorá smeruje smerom dole. Z aktuálnej pozície čítacej hlavice sa vykoná špeciálny automat označený ako *S*. Ten určí, ako ďaleko sa dá dostať smerom doprava. Pri štvorke to je oveľa viac, ako pri jednotke.

Ak *S* nerozpozná nič, výsledkom je číslo 1. V opačnom prípade sa postupuje do uzlu, ktoré má označenie Podmienka 5



Obr. 4.11: Štvrtý uzol

Podmienka 5

Aktuálnu hodnotu čítacej hlavice si označme ako bod B. Bod A bude pozícia čítacej hlavice pred prevedením *Automatu A*. Bod R bude počiatočný bod, kde sa začínalo rozpoznávanie. Pre podmienku s označením 5 platí, nasledovné:

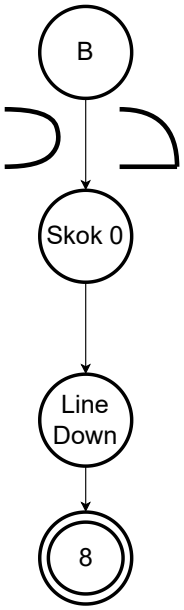
- Vypočíta sa vzdialenosť |AR| medzi bodmi A a R.
- Vypočíta sa vzdialenosť |BA| medzi bodmi B a A.
- Ak je vzdialenosť |AR| oveľa väčšia ako vzdialenosť |BA| → našlo sa číslo 1.
- Ináč sa pokračuje ďalej, pravdepodobne sa našlo číslo 4, ale treba to ešte overiť.

Pokračuje sa ďalej ako je to zobrazené na obrázku 4.11. Čítacia hlavica sa presunie na novú pozíciu, kde skončil *Automat A*. Posunie sa o pár symbolov nižšie. Spustí sa posledný automat *Line Right*. Posúva sa smerom doprava, a hľadá hrany (zmena symbolov z ”1” → ”0” a ”0 → ”1”). Aby sa číslica štyri rozpoznala, musia sa nájsť práve 2 hrany, ktoré reprezentujú takzvanú "nožičku"smerom dole. Týmto spôsobom sa rozpoznala číslica 4.

Číslo 8

Pri poslednej piatej vetve sa výber čísla zúžil iba na jediného kandidáta a to je číslo 8. Výsledkom predchádzajúceho automatu musela byť šikmá a vertikálna čiara.

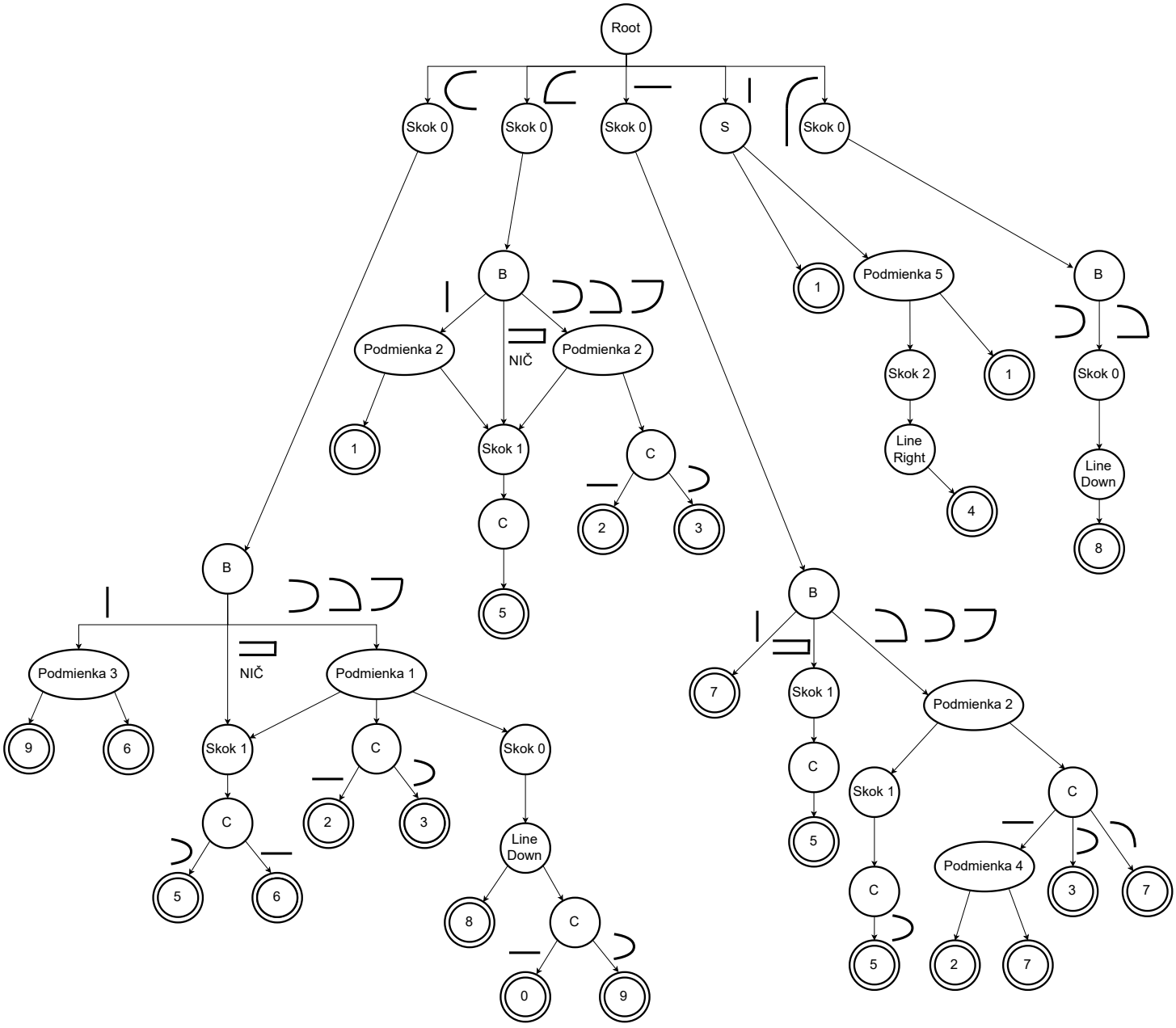
Na prvý pohľad nemusí byť jasné, prečo práve aj vertikálna čiara. Najčastejšie to nastáva v prípade, že číslo, ktoré sa rozpoznáva, je hrubšie. Tým, že druhá časť *automatu A* (3.3.1) má primárny pohyb smerom dole, môže sa stať, že nebude presne opisovať hranu čísla. Pôjde smerom dole až na spodnú časť čísla 8.



Obr. 4.12: Piaty uzol

Po skoku na pôvodnú pozíciu spustí *Automat B* ako je to na obrázku 4.12 a kontroluje sa pravá časť objektu. Ak sa dá pokračovať (rozpoznali sa 2 krát šikmá čiara, alebo šikmá a horizontálna čiara), čítacia hlavica sa opäť presúva na pôvodnú pozíciu a spustí sa automat *Line Down*. Ten spočíta počet hrán, ako už bolo spomenuté vyššie. V prípade úspechu sa rozpoznalo číslo 8.

Na obrázku 4.13 je zobrazený celý graf, ktorý bol opísaný v tejto kapitole. Pri každom rozpoznávaní čísla sa prechádza v tomto grafe od hora (uzol Root) smerom nadol ku listovým uzlom.



Obr. 4.13: Celkový graf

Kapitola 5

Implementácia

Táto kapitola a jej podkapitoly sa zaoberajú implementáciou výslednej aplikácie na rozpoznávanie čísel z obrázka. Využíva sa pri tom dvojrozmerný skákajúci automat. Vysvetlený je tu postup ako sa tvorili jednotlivé automaty, ako boli vyriešené niektoré problémy pri implementácii.

Všetky automaty a rozhodovací strom z predchádzajúcich kapitol sú implementované v jazyku C++. Postupne sa prehľadáva celý obrázok a hľadajú sa všetky čísla. Pre jednoduchšiu prácu bol zvolený framework Qt.

5.1 Použité prostriedky

Qt

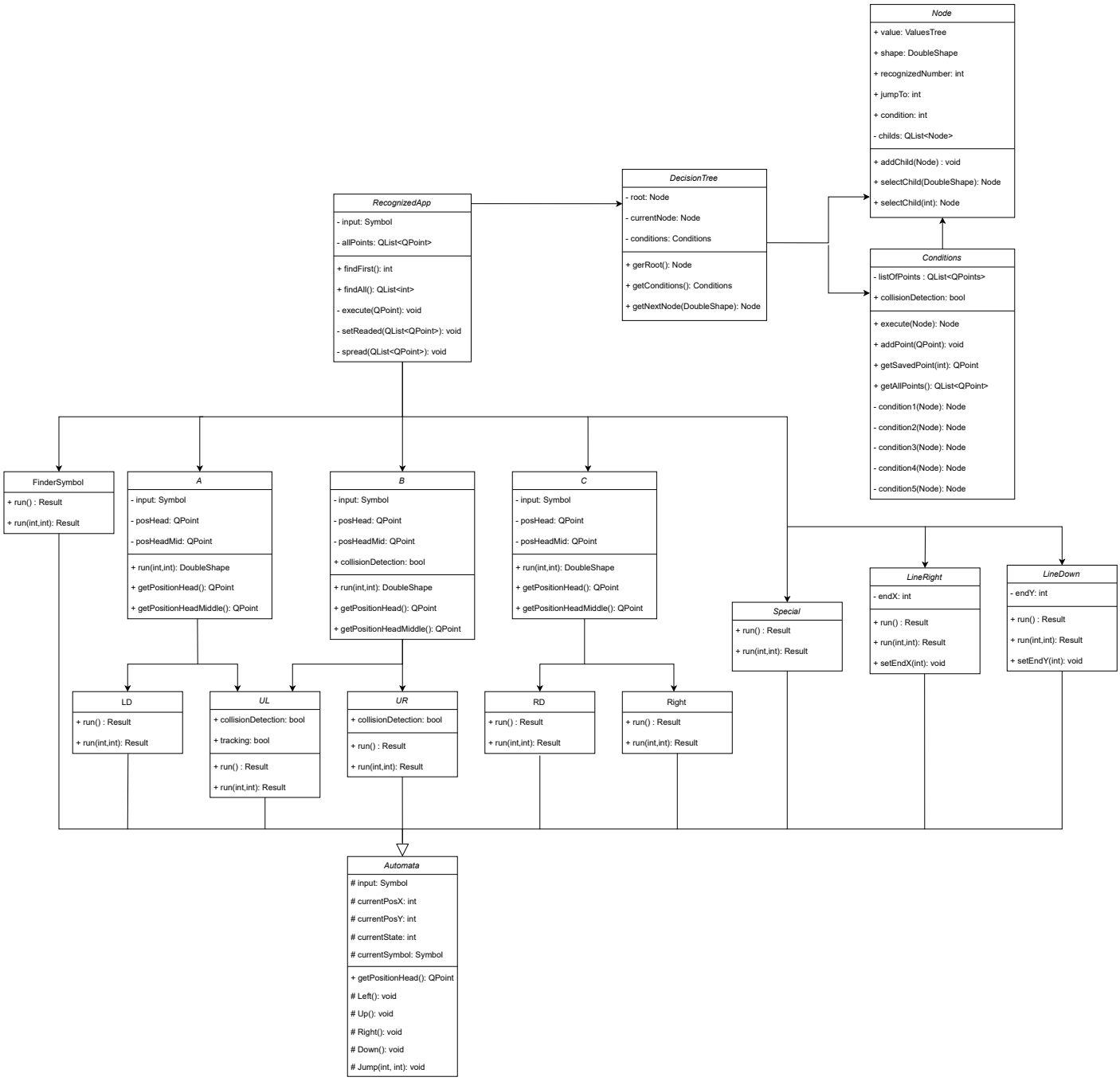
Qt je súprava nástrojov pre vytvorenie grafických používateľských rozhraní. Fungujú pod rôznymi operačnými systémami ako sú Linux, Windows, macOS, Android. Existuje implementácia pre jazyk C++ ako aj Python (PyQt).

Framework Qt je dodávaný spoločne s vlastným vývojovým prostredím Qt Creator. [8]

QMake

Nástroj qmake pomáha zjednodušiť proces zostavovania vývojových projektov na rôznych platformách. Automatizuje generovanie súborov Makefile, takže na vytvorenie každého súboru Makefile je potrebných len niekoľko riadkov informácií. qmake sa môže použiť pre akýkoľvek softvérový projekt, či už je napísaný s Qt alebo nie. [8]

5.2 Diagram tried



Obr. 5.1: Diagram tried

Na obrázku 5.1 je diagram tried súvisiaci s reprezentáciou objektov v scéne. Zobrazuje vzťahy medzi jednotlivými objektami, ktoré reprezentujú časť aplikácie na rozpoznávanie čísel.

5.3 Načítanie vstupu

Program dostane na vstupe obrázok. Ten môžme zvoliť, keď vyberieme jeho umiestnenie. Tiež sa dá vybrať aj celý priečinok. Program tak bude v ňom hľadať všetky súbory s príponami *jpg*, alebo *png*. Každý jeden bude postupne načítavať a spúšťať na ňom implementované automaty.

Po úspešnom načítaní obrázka sa vyberú jednotlivé hodnoty pixelov (RGB hodnota). Následne sa prevedú do reprezentácie o troch hodnotách:

- *Border* - Hraničný symbol. Nachádza sa na okraji celého vstupu.
- *Black* - Čierny symbol. Reprezentuje farbu čísla, tmavé pixely, pre ktoré platí, že ich každá zložka (Red, Green, Blue) je väčšia ako prahová hodnota 169.
- *White* - Biely symbol. Reprezentuje farbu okolia. Sú to všetky ostatné pixely, ktoré nesú hraničné, ani súčasťou čísla.

Pri rozpoznaní jedného obrázka sa výsledok objaví na pravej časti aplikácie. V prípade, že sa rozpoznávanie spustilo nad priečinkom, vygeneruje sa záznam (*log.txt*), ktorý bude mať nasledovný formát:

[Názov priečinka]
[obrázok 1] [rozpoznané číslo]
[obrázok 2] [rozpoznané číslo]
.
.
.
[obrázok n] [rozpoznané číslo]
Zhrnutie
Počet 0: [počet čísel]
.
.
.
Počet 9: [počet čísel]
Celkový počet obrázkov: [číslo]

5.4 Automaty

Je tu vytvorená základná trieda *Automata*. Ako jediný parameter berie vstupnú pásku. Tá je tvorená symbolmi, ktoré boli opísané vyššie 5.3 v časti načítanie vstupu.

Obsahuje metódy na pohyb *Left*, *Right*, *Up* a *Down*, ktoré posúvajú aktuálnu hodnotu čítacej hlavice do daného smeru. Tiež je tu aj metóda *Jump*, ktorá vyžaduje súradnice x a y na presun čítacej hlavice na zvolenú pozíciu.

Z tejto triedy dedia všetky ostatné implementované automaty. Každý má svoju vlastnú metódu *run*, ktorou sa spustí. Ak sa nezadajú parametre pozície x a y, spúšťa sa z miesta, kde skončil posledný beh tohto automatu. V prípade prvého behu sa začína na súradniciach $x = 1$ a $y = 1$. Je to prvý symbol v ľavej hornej časti vstupu.

5.4.1 Hľadanie začiatku rozpoznávania

Tento automat slúži na nájdenie prvého čierneho symbolu. Ten môže byť potencionálne časťou objektu, v našom prípade číslice, ktorú sa snažíme rozpoznať. Pohybuje sa zľava doprava po riadkoch. Po spustení metódy *run* sa vykonáva nasledovné:

- Ak sa narazí na biely symbol, poprípade na symbol označený ako prečítaný (automat cez tento symbol už raz prešiel), pohybuje sa smerom doľava.
- V prípade, že narazí na symbol hranice, posunie sa o jeden riadok nižšie. V praxi to znamená, že sa využije skok, kedy sa presunie čítacia hlavica na začiatok nového riadku. Postupne sa tak prehľadávajú všetky symboly.
- Ak sa nájde čierny symbol prehľadávanie končí úspešne, vráti sa hodnota *true*. V premennej *PositionHead* sa tak nájde aktuálna pozícia.
- Posledný prípad, čo môže nastať je ten, že automat prejde celý vstup a už nenájde žiaden čierny symbol. Pri skoku na nový riadok sa na prvej pozícii nájde hneď hraničný symbol. Znamená to, že sa rozpoznávanie už skončilo. Prehľadal sa celý vstup. Vráti sa hodnota *false*.

Ak je vstupný obrázok veľký (má veľké rozlíšenie), táto časť môže trvať dlhšie. Urýchliť sa dá tým, že by sa pri presune na nový riadok neposúvalo iba po jednom riadku. Nato slúži v kóde premenná *step*, ktorá určuje o koľko riadkov sa posunie (1 znamená, že sa bude prehľadávať každý riadok).

Hodnota *step* nemôže byť príliš veľká. Potom by sa mohlo stať, že sa budú preskakovať číslice, poprípade sa zle vyhodnotí výsledná číslica.

5.4.2 Jednoduché

To ako vyzerajú jednotlivé automaty je popísané v sekcii 3.3 o rozpoznávaní čísel. Pre všetky jednoduché automaty platí to, že ich výsledkom je objekt *Result*. Obsahuje nasledovné:

- Tvar
- sizeX
- sizeY

Tvar značí útvar, ktorý sa rozpoznal. Môže to byť horizontálna čiara (LineH), vertikálna čiara (LineV), šikmá čiara (Diagonal), alebo nemusí sa rozpoznať nič (None).

Parametre *sizeX* a *sizeY* určujú dĺžku rozpoznaného útvaru v osi x a osi y. Sú užitočné pri ďalšej práci s týmito objektami. Napríklad, ak sú príliš malé, môžme ich zanedbať. Výsledný útvar sa označí na *None* a pokračuje sa ďalej ako keby automat skončil v stave, v ktorom sa tvar nerozpoznal.

Automat UL

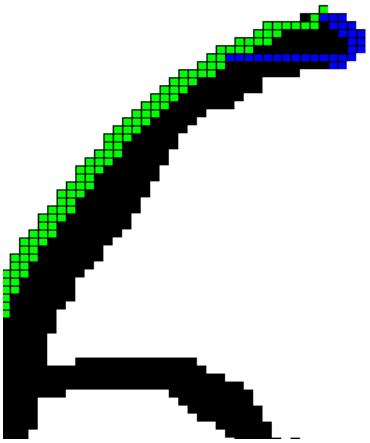
Teoreticky je tento automat popísaný v predchádzajúcej sekcii 3.3.1. Pohybuje sa smerom dole a doľava. Primárny pohyb je doľava. Nachádzajú sa tu dva koncové stavy. *FState* je stav, do ktorého sa dostane automat v prípade, že sa nič nerozpoznalo. Výsledkom je útvar *None*.

TState je opačný stav, kedy je rozpoznaný určitý tvar. Pre výsledok musí platiť nasledovné:

- LineV - tento útvar sa rozpozná vtedy, ak je hodnota premennej *sizeX* približne rovnako veľká ako *sizeY*, čiže rozdiel týchto hodnôt je maximálne 20%. V praxi to znamená, že sa vypočíta minimálna a maximálna hodnota *sizeX*. Následne musí platiť, že *sizeY* sa nachádza medzi týmito novo vypočítanými hodnotami.
- LineH - tento útvar sa rozpozná vtedy, keď hodnota *sizeX* je oveľa väčšia ako *sizeY*. V praxi to znamená, že sa hodnota *sizeX* zmenší o 70% a musí byť stále väčšia ako *sizeY*.
- Diagonal - ak neplatia dve predchádzajúce možnosti, výsledkom je práve šikmá čiara.

V hlavnej slučke, ktorá sa vykonáva, kým sa nevkročí do koncového stavu, sa nachádza detekcia prečítaných symbolov. Je to klasická podmienka. Kontroluje sa tu, či sa neprečítal už prečítaný symbol a súčasne musí platiť, že sa automat nenachádza v počiatočnom stave. Je to implementované z dôvodu lepšieho určenia výslednej číslice 5 a 6.

Taktiež sa tu nachádza aj premenná *tracking*. Keď je nastavená na hodnotu *true*, tak sa každý čierny symbol cez ktorý sa prejde zmení na prečítaný. Je to implementované pre lepšie určenie výslednej číslice 5 a 6.



Obr. 5.2: Ukážka kolízie

Na obrázku 5.2 je znázornená ukážka jedného z prípadov, kedy sa nastaví hodnota kolízie na true. Zelená farba ukazuje pohyb *Automatu A* a modrá znázorňuje pohyb *Automatu B*.

Automat LD

Teoreticky je tento automat popísaný v predchádzajúcej sekcii 3.3.1. Pohybuje sa smerom dole a doprava. Primárny pohyb je dole. Nachádzajú sa tu dva koncové stavy. *FState* je stav, do ktorého sa dostane automat v prípade, že sa nič nerozpoznalo. Výsledkom je útvar *None*.

TState je opačný stav, kedy je rozpoznaný určitý tvar. Pre výsledok musí platiť nasledovné:

- LineH - tento útvar sa rozpozná vtedy, ak je hodnota premennej *sizeX* približne rovnako veľká ako *sizeY*, čiže rozdiel týchto hodnôt je maximálne 20%. V praxi to znamená, že sa vypočíta minimálna a maximálna hodnota *sizeY*. Následne musí platiť, že *sizeX* sa nachádza medzi týmito novo vypočítanými hodnotami.
- LineV - tento útvar sa rozpozná vtedy, keď hodnota *sizeY* je oveľa väčšia ako *sizeX*. V praxi to znamená, že sa hodnota *sizeY* zmenší o 70% a musí byť stále väčšia ako *sizeX*.
- Diagonal - ak neplatia dve predchádzajúce možnosti, výsledkom je práve šikmá čiara

Automat UR

Teoreticky je tento automat popísaný v predchádzajúcej sekcii 3.3.1. Pohybuje sa smerom doprava a dole. Primárny pohyb je doprava. Nachádzajú sa tu dva koncové stavy. *FState* je stav, do ktorého sa dostane automat v prípade, že sa nič nerozpoznalo. Výsledkom je útvar *None*.

TState je opačný stav, kedy je rozpoznaný určitý tvar. Pre výsledok musí platiť nasledovné:

- LineV - tento útvar sa rozpozná vtedy, ak je hodnota premennej *sizeX* približne rovnako veľká ako *sizeY*, čiže rozdiel týchto hodnôt je maximálne 20%. V praxi to znamená, že sa vypočíta minimálna a maximálna hodnota *sizeY*. Následne musí platiť že *sizeX* sa nachádza medzi týmito novo vypočítanými hodnotami.
- LineH - tento útvar sa rozpozná vtedy, keď hodnota *sizeX* je oveľa väčšia ako *sizeY*. V praxi to znamená, že sa hodnota *sizeX* zmenší o 70% a musí byť stále väčšia ako *sizeY*.
- Diagonal - ak neplatia dve predchádzajúce možnosti, výsledkom je práve šikmá čiara

Takisto aj v tomto automate sa nachádza detekcia prečítaných symbolov. Je to klasická podmienka. Kontroluje sa tu, či sa neprečítal už prečítaný symbol a súčasne musí platiť, že sa automat nenachádza v počiatočnom stave. Je to implementované z dôvodu lepšieho určenia výslednej číslice 5 a 6.

Ukážka ako môže nastať takáto kolízia je zobrazená na obrázku 5.2.

Automat RD

Teoreticky je tento automat popísaný v predchádzajúcej sekcii 3.3.1. Pohybuje sa smerom dole a doľava. Primárny pohyb je dole. Nachádzajú sa tu dva koncové stavy. *FState* je stav, do ktorého sa dostane automat v prípade, že sa nič nerozpoznalo. Výsledkom je útvar *None*.

TState je opačný stav, kedy je rozpoznaný určitý tvar. Pre výsledok musí platiť nasledovné:

- LineV - tento útvar sa rozpozná vtedy, ak je hodnota premennej *sizeX* približne rovnako veľká ako *sizeY*, čiže rozdiel týchto hodnôt je maximálne 20%. V praxi to znamená, že sa vypočíta minimálna a maximálna hodnota *sizeX*. Následne musí platiť, že *sizeY* sa nachádza medzi týmito novo vypočítanými hodnotami.
- LineH - tento útvar sa rozpozná vtedy, keď hodnota *sizeX* je oveľa väčšia ako *sizeY*. V praxi to znamená, že sa hodnota *sizeX* zmenší o 70% a musí byť stále väčšia ako *sizeY*.
- Diagonal - ak neplatia dve predchádzajúce možnosti, výsledkom je práve šikmá čiara

5.4.3 Ostatné

V tejto sekcii je v skratke opísaná implementácia *Spojených* a *Špeciálnych* automatov.

Spojené

Sú to triedy, ktoré vo svojej funkcii *run* zavolajú postupne dva jednoduché automaty a spoja ich výsledok do jednej štruktúry. Je to dvojica výsledných útvarov v tvare, kde *Útvar1* je výsledok prvého automatu a *Útvar2* je výsledok druhého:

(Útvar1,Útvar2)

Automat A

Teoreticky je popísaný v predchádzajúcej sekcii 3.3.2. Spustí sa automat UL a nastaví sa jeho parameter *tracking* na hodnotu *true*. Tým sa budú meniť všetky čierne symboly za prečítané, cez ktoré prejde.

Ak jeho výsledok je vertikálna čiara (LineV), ďalší automat sa už nespustí a výsledok je dvojica (LineV,None). Je to implementované tak z dôvodu rýchlejšieho rozpoznania.

To isté platí aj pri rozpoznanom útvare horizontálna čiara (LineH). Výsledkom je potom dvojica (LineH,None).

Pri ostatných prípadoch sa spustí aj nasledujúci automat LD z miesta, na ktorom skončil UL. Tesne pred koncom sa ešte kontroluje veľkosť rozpoznaného útvaru. Ak aj bol rozpoznaný, ale je príliš malý (hodnoty sizeX a sizeY sú menšie ako 10), zmení sa na hodnotu *None*.

Automat B

Teoreticky je popísaný v predchádzajúcej sekcii 3.3.2. Spustí sa automat UR.

Ak jeho výsledok je vertikálna čiara (LineV), ďalší automat sa už nespustí a výsledok je dvojica (LineV,None). Je to implementované tak z dôvodu rýchlejšieho rozpoznania.

Nasleduje ďalší automat UL z miesta, na ktorom skončil UR. Parameter *tracking* bude v tomto prípade vypnutý (hodnota $false$). Výsledky oboch častí sa spoja do jednej štruktúry.

Tesne pred koncom sa ešte kontroluje, veľkosť rozpoznaného útvaru. Ak aj bol rozpoznaný, ale je príliš malý (hodnoty sizeX a sizeY sú menšie ako 10), zmení sa na hodnotu *None*.

Automat C

Teoreticky je popísaný v predchádzajúcej sekcii 3.3.2. Spustí sa automat *Right*.

Ak jeho výsledok je horizontálna čiara (LineH), ďalší automat sa už nespustí a výsledok je dvojica (LineH,None). Je to implementované tak z dôvodu rýchlejšieho rozpoznania. Využíva sa to najmä pri čísle 2.

V opačnom prípade sa pokračuje s automatom *RD* z miesta, kde skončil predchádzajúci automat *Right*. Výsledky oboch častí sa spoja do jednej štruktúry.

Tesne pred koncom sa ešte kontroluje veľkosť rozpoznaného útvaru. Ak aj bol rozpoznaný, ale je príliš malý (hodnoty sizeX a sizeY sú menšie ako 10), zmení sa na hodnotu *None*.

Špeciálne

Zabezpečujú špecifickú činnosť, ktorá vedie k lepšiemu rozoznaniu niektorých čísel. Dedia z triedy *Automata*, takže sa taktiež môžu pohybovať po vstupnej páske. Výsledkom týchto automatov je rovnaká štruktúra ako majú vyššie zmienené Spojené automaty. Je to z dôvodu, aby sa dalo pracovať s oboma typmi rovnaným spôsobom.

Line Down a Line Right

Metóda *run* obsahuje implementáciu daného automatu, ktorý bol bližšie popísaný v minulej kapitole 3.3.3.

Ak je hodnota premennej *countSwap* (počet detekovaných hrán) menšia alebo rovná hodnote 3, výsledná štruktúra je (LineH,None). V opačnom prípade je to (None,None).

Podobne je to aj pri automate *Line Right*, avšak hodnota premennej *countSwap* (počet detekovaných hrán) musí byť 2, alebo 3, aby bola výsledná štruktúra (LineH,None).

Automat S

Aj tu sa vyberá výsledok z dvoch hodnôt. Ak sa čítacia hlavica pohla o viac ako 10 symbolov, výsledná štruktúra je (LineH,None). Ináč sa je to (None,None).

5.5 Rozhodovací strom

Rozhodovací strom je tvorený niekoľkými triedami. Základná trieda *Node* reprezentuje jeden uzol v strome. Každý takýto uzol obsahuje nasledovníka na ďalší uzol. Ďalej je tu trieda *Conditions*. Tá obsahuje všetky podmienky, na základe ktorých sa potom vyberá v niektorých prípadoch určitý uzol. Všetko to zaobaľuje trieda *DecisionTree*, ktorá poskytuje jednoduchú prácu s touto štruktúrou.

5.5.1 Strom

Uzol pozostáva z nasledujúcich častí:

- Id - Jednoznačný identifikátor uzla
- Value - Hodnota, ktorá reprezentuje aký automat sa spustí, poprípade, či sa vykoná skok alebo určitá podmienka. Môže obsahovať jednu z nasledujúcich hodnôt:
  - None - ak je prázdna
  - A,B,C LineDown, LineRight, Special - jednu z hodnôt. ktorá reprezentuje automat, ktorý sa spustí
  - Jump - pre skok
  - Condition - pre podmienku
  - RecognizedNumber - v prípade, že je to listový uzol
- Shape - Tvar, ktorý musí byť rozpoznaný, aby sa dalo dostať do tohto uzla. Používa sa pri vyberaní ďalšieho nasledovníka.
- RecognizedNumber - Nastaví sa iba v prípade, že daný uzol je listový. Hodnota potom reprezentuje, aké číslo sa rozpoznalo.
- JumpTo - Obsahuje číslo (identifikátor) pozície, do ktorej sa presunie čítacia hlavica. Nastavuje sa iba v prípade, že sa bude skákať.
- Condition - Ak sa bude vykonávať podmienka, v tejto premennej je uložené, ktoré číslo (identifikátor) podmienky sa vykoná.
- Childs - zoznam potomkov (ukazovateľov na uzly), do ktorých sa dá dostať z aktuálneho uzla.

Táto trieda obsahuje aj metódy, jedna z najzaujímavejších je metóda *SelectChild*. Pri jej zavolaní sa vyberie ďalší nasledovník a to buď na základe indexu (číslo reprezentujúce pozíciu daného uzlu), alebo konkrétneho útvaru (hodnota Shape), ktorý musí daný uzol obsahovať. Ak sa nenájde požadovaný uzol, vráti sa hodnota *null*. Tým sa aj končí rozpoznávanie. Výsledné číslo sa nenašlo.

DecisionTree

Nachádza sa tu trieda s podmienkami *Conditions*. Tie sa hneď aj vyhodnotia. Uložený je tu ukazovateľ na koreňový uzol stromu, aby sa dalo kedykoľvek vrátiť na začiatok. Tiež je tu uložený aktuálny uzol stromu, v ktorom sa program nachádza. V konštruktory sa vytvorí celý strom.

Najdôležitejšou častou tejto triedy je metóda *GetNextNode*. Ako jediný parameter obsahuje útvar, ktorý bol rozpoznaný. Z aktuálneho uzla sa zavolá metóda *SelectChild*. Tá vyberie na základe vstupného parametru nasledujúci uzol a uloží jeho ukazovateľ do aktuálneho uzlu. Na konci sa ešte skontroluje, či uzol neobsahuje podmienku. V prípade, že áno, podmienka sa vyhodnotí a metóda vráti výsledný uzol.

5.5.2 Podmienky

Vyhodnocujú sa tu podmienky. Volá sa tu metóda *execute*, ktorá vyberá jednu z piatich podmienok. Nižšie budú všetky bližšie opísané. Ako vstupným parametrom je uzol, ktorý obsahuje podmienku. Výstupom je nasledujúci potomok.

Nachádza sa tu zoznam všetkých bodov (*listOfPoints*), v ktorých skončilo rozpoznávanie jednotlivých automatov, ktoré sa spustili. Je to potrebné z viacerých dôvodov. Jedným z nich sú skoky. Je potrebné vedieť, kam sa má skočiť. Skáče sa iba na miesta, kde sa automat už v minulosti nachádzal. Ďalším dôvodom je porovnávanie, napríklad ako ďaleko sa pohla pozícia čítacej hlavice. Ak si pri rozpoznaní čísla alebo hocijakého iného objektu nie sme si úplne istí aký to je, dá sa využiť tento zoznam bodov.

Body sa pridávajú na koniec zoznamu pomocou metódy *AddPoint*. Tá má ako parameter súradnice x a y bodu, ktorý sa má uložiť.

Condition 1

Uzol, na ktorom sa spustí táto podmienka, musí mať presne troch nasledovníkov. Z nich sa vyberie potom jeden. Viac popísaná táto podmienka a jej použitie sa nachádza v predchádzajúcej kapitole 4.2.1.

Zo zoznamu uložených bodov sa vyberú tie, ktoré majú nasledovný index.

- 0 (ďalej označená ako R) - počiatočná hodnota začiatku
- 2 (ďalej označená ako A) - po prevedení *automatu A*
- 4 (ďalej označená ako B) - po prevedení *automatu B*

Ak je hodnota B (zmenšená o 10%) väčšia ako A, vyberie sa prvý nasledovník. Počíta sa tu s miernou odchýlkou.

Ak rozdiel medzi hodnotami A a R oveľa väčší ako rozdiel medzi hodnotami B a R, vyberie sa druhý nasledovník. Implementované je to tak, že sa výsledok rozdielu A a R zmenší o 70%. Rovnaký nasledovník sa vyberie aj keď bola detekovaná kolízia.

Ak neplatí ani jedno z predchádzajúcich, vyberie sa posledný tretí nasledovník.

Condition 2

Uzol, na ktorom sa spustí táto podmienka, musí mať dvoch nasledovníkov. Z nich sa vyberie potom jeden. Viac popísaná táto podmienka a jej použitie sa nachádza v predchádzajúcej kapitole 4.2.2.

Zo zoznamu uložených bodov sa vyberú tie, ktoré majú nasledovný index.

- 2 (ďalej označená ako A) - po prevedení *automatu A*
- 4 (ďalej označená ako B) - po prevedení *automatu B*

Podobne ako v predchádzajúcej podmienke, aj tu sa porovnajú hodnoty A a B. V prípade, že je hodnota B väčšia ako hodnota A, vráti sa prvý nasledovník. Ináč sa vráti druhý.

Condition 3

Uzol, na ktorom sa spustí táto podmienka, musí mať dvoch nasledovníkov. Z nich sa vyberie potom jeden. Viac popísaná táto podmienka a jej použitie sa nachádza v predchádzajúcej kapitole 4.2.1.

Zo zoznamu uložených bodov sa vyberú tie, ktoré majú nasledovný index.

- 2 (ďalej označená ako A) - po prevedení *automatu A*
- 4 (ďalej označená ako B) - po prevedení *automatu B*

Táto podmienka je takmer rovnaká ako predchádzajúca podmienka 2, avšak s malým rozdielom. Pre bod A musí platiť že je väčší ako bod B, aj vtedy, keď bod A bude zmenšený o 30%, potom sa vráti druhý nasledovník. Inak sa vráti prvý.

Condition 4

Uzol, na ktorom sa spustí táto podmienka, musí mať dvoch nasledovníkov. Z nich sa vyberie potom jeden.

Zo zoznamu uložených bodov sa vyberú tie, ktoré majú nasledovný index

- 0 (ďalej označená ako R) - počiatočná hodnota začiatku
- 5 (ďalej označená ako C) - po prevedení *automatu C*

Ak je hodnota R (zväčšená o 10%) väčšia ako C, vyberie sa druhý nasledovník. Počíta sa tu s miernou odchýlkou.

Ináč sa vyberie prvý nasledovník.

Condition 5

Uzol, na ktorom sa spustí táto podmienka, musí mať dvoch nasledovníkov. Z nich sa vyberie potom jeden. Viac popísaná táto podmienka a jej použitie sa nachádza v predchádzajúcej kapitole 4.2.4.

Zo zoznamu uložených bodov sa vyberú tie, ktoré majú nasledovný index.

- 0 (ďalej označená ako R) - počiatočná hodnota začiatku
- 2 (ďalej označená ako A) - po prevedení *automatu A*
- 3 (ďalej označená ako B) - po prevedení prvej časti *automatu B*

Ak je rozdiel súradníc y medzi bodmi A a R (ďalej iba |AR|) menší ako rozdiel súradníc x medzi bodmi B a A (ďalej iba |BA|), vyberie sa druhý nasledovník.

Ak je |AR| (zmenšený o 70%) väčší ako |BA|, vyberie sa prvý nasledovník.

5.5.3 Zhrnutie

Trieda *RecognizedApp* pracuje s vyššie uvedenou stromovou štruktúrou. Pre nájdenie objektov v obrázku sa volá metóda *findFirst* alebo *findAll*. Tá prvá nájde objekt, ktorý sa pokúsi rozpoznať a vráti rozpoznané číslo, poprípade -1 keď sa nenašlo nič. *findAll* postupne prehľadá celý vstupný obrázok a hľadá v nich čísla. Vráti zoznam všetkých čísel, ktoré sa nachádzajú vo vstupnom obrázku.

Automaty, ktoré sa spúšťajú a rozpoznávajú tak číslo, neprechádzajú cez všetky symboly. Skáču po vstupnej páske, aby vykonali svoju prácu, čo najrýchlejšie. Tým, že sa v priebehu rozpoznávania nemenia čierne symboly na vstupnej páske za prečítané, musí sa to urobiť na konci. Je to potrebné kvôli tomu, aby sa viackrát neprehľadával ten istý objekt.

Existuje viacero spôsobov ako to docieliť. Mohli by sa nájsť maximálne a minimálne hodnoty súradníc x a y všetkých bodov, cez ktoré automaty prechádzali. Následne by všetky body medzi nimi sa označili ako prečítané. Výhodu to má tú, že to nie je veľmi časovo náročné. Problém je ale v tom, že niektoré časti čísla sa nezakryjú prečítanými symbolmi. Pri ďalšom prechode sa tak môžu objaviť nové útvary, ktoré sa na začiatku nenachádzali na vstupe.

Nakoniec je použitá metóda, ktorá zaberie viac času, ale vo väčšine prípadov zakryje celý číselný útvar. Vyberú sa niektoré symboly, ktoré patria do obrázka a uložia sa do zoznamu. Následne sa na ne aplikujú jednoduché pravidlá:

- Vyberie sa prvý symbol zo zoznamu sa nastaví ako prečítaný.
- Skontrolujú sa všetky 4 smery okolo daného symbolu. Ak sa okolo neho nachádza ďalší čierny symbol, pridá sa na koniec do zoznamu.
- Opakuje sa to, kým zoznam nie je prázdny.

Tak sa docieli, že všetky symboly sa nastavia ako prečítané. Problém nastáva vtedy, ak rozpoznané číslo nie je spojité. Táto metóda sa aplikuje po každom nájdenom aj nenájdenom objekte.

Existujú aj iné metódy, ktoré sú efektívnejšie, ale pre účel tejto aplikácie to stačí.

Kapitola 6

Testovanie

V tejto kapitole je opísané testovanie vyššie uvedeného programu na rozpoznávanie čísel z obrázka. Program podporuje rozpoznanie aj viacerých čísel. Testovanie bolo hlavne zamerané na rozpoznávanie jedného čísla.

Existujú rôzne premenné, ktoré sa dajú meniť a tak testovať, či už na presnosť, alebo rýchlosť nájdenia jednotlivých čísel. Napríklad rôzne varianty automatu *FinderSymbol*. Pomôže to k rýchlejšiemu nájdeniu čísla.

Dajú sa meniť jednoduché automaty tak, aby dokázali rozpoznávať čiary ináč. Takisto je možné meniť parametre, ktoré určujú od kedy začína byť rozpoznaná horizontálna čiara, vertikálna, poprípade šikmá.

Jedným z dôležitých faktorov je aj prahová hodnota, ktorá kontroluje, či je daný pixel biely, alebo čierny, poprípade či patrí do okolia alebo je súčasťou čísla.

Na účel tohto testovania som sa rozhodol, že otestujem iba niektoré varianty, keďže všetkých možností je veľmi veľa.

6.1 Testovanie na existujúcom datasete

MNIST

Databáza MNIST je (anglicky: Modified National Institute of Standards and Technology) je veľká databáza ručne písaných číslic, ktorá sa používa v rôznych systémoch na spracovanie obrazu.

Pôvodné čiernobiele (dvojúrovňové) obrázky z databázy NIST boli normalizované tak, aby sa zmestili do poľa 20 x 20 pixelov pri zachovaní ich pomeru strán. Výsledné obrázky majú vyhladené hrany a boli vycentrované na obrázku 28 x 28 pixelov.

Databáza bola vytvorená zmiešaním vzoriek z pôvodných datasetov NIST. Autori poskytujú na svojich stránkach celú trénovaciu sadu, ktorá je tvorená z 60 000 vzoriek a testovaciu sadu, ktorá pozostáva z 10 000 vzoriek. [4]

MNIST je teda databáza čísel, ktorá bola využitá pre testovanie. Jedná sa o databázu amerického písma. Na účel tohto testovania bola podľa potrieb upravená. Neboli použité všetky vzorky (obrázky čísel). Boli vybraté také, ktoré sa najviac podobali tým našim európskym. Neboli použité obrázky, ktoré boli príliš zdeformované, alebo čísla neboli napísané v celku.

Pre každú číslicu 0-9 bolo vybraných okolo 100-150 vzoriek z daného datasetu. Vybrané vzorky mali mali veľkosť 28 x 28 pixelov. Pre implementovaný automat to boli príliš malé obrázky. Na trénovanie a testovanie neurónových sieti taká veľkosť postačuje, ale pre tento účel nie. Preto boli všetky obrázky zväčšené na 300 x 300 pixelov. Zväčšenie bolo prevedené online nástrojom, ktorý upravil všetky vzorky na požadovanú veľkosť. Veľkosť je dostatočná na správne fungovanie implementovaných automatov a zároveň ich veľkosť nezaberá príliš veľa miesta.

6.1.1 Úspešnosť na datasete MNIST

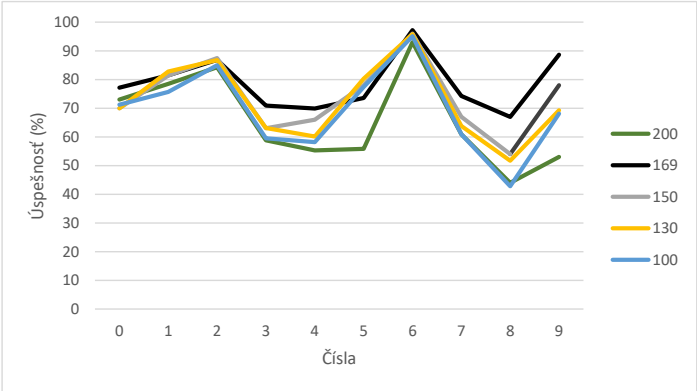
číslo	úspešnosť %	počet čísel	0	1	2	3	4	5	6	7	8	9
0	77.1242	153	118	0	2	0	0	0	26	0	2	0
1	81.4286	70	0	57	0	0	2	0	1	0	1	0
2	86.8750	160	3	2	139	2	1	4	0	1	0	4
3	70.9220	141	3	2	4	100	2	6	2	1	6	8
4	69.9029	103	0	1	2	0	72	8	5	0	0	0
5	77,451	102	0	0	0	0	0	79	11	0	0	0
6	97,2028	143	0	0	0	0	2	1	139	0	0	0
7	74,2857	105	0	1	5	1	0	8	0	78	0	4
8	66.9643	112	2	0	4	0	0	14	5	0	75	4
9	88.6364	88	0	0	1	0	0	7	1	0	0	78

Obr. 6.1: Výsledky testovania

Na obrázku 6.1 je zobrazená tabuľka testovania všetkých čísel. Všetky parametre boli nastavené tak, ako boli opísané v implementácii. Prvý stĺpec reprezentuje testované číslo. Ďalší stĺpec zobrazuje úspešnosť rozpoznania jednotlivého čísla. Stĺpec s názvom počet čísel obsahuje hodnotu, na koľkých číslach sa rozpoznávalo. Ostatné stĺpce ukazujú koľkokrát sa našlo ktoré číslo. V niektorých prípadoch sa stalo, že sa číslo na obrázku vôbec nerozpoznalo.

Pri 0 sa viackrát stalo, že sa našla číslica 6. Pri jednotke sa väčšinou rozpoznávala iba čiara smerom dole. 2, 6 a 9 majú už pomerne dobrú úspešnosť. Mali málo chýb. Najhoršie dopadli čísla 4 a 8. Pri štvorke bolo potrebné, aby sa rozpoznala najprv čiara smerom dole. Pri väčšom náklone čísla to už prestávalo platiť a to mohol byť jeden z dôvodov nižšej úspešnosti. Osmička bola jedným z najnáročnejších čísel na rozpoznanie. Ak bola veľmi sploštená, mýlila sa s číslom 5.

6.1.2 Prahové hodnoty



Obr. 6.2: Graf prahových hodnôt

Graf 6.2 zobrazuje ako sa líšia jednotlivé úspešnosti čísel pri rôznej prahovej hodnote (hodnota, ktorá určí či ide o biely, alebo čierny symbol). Na x-ovej ose sú jednotlivé čísla 0-9, ktoré boli rozpoznané. Os y udáva úspešnosť jednotlivých čísel. Hodnoty patriace jednej prahovej hodnote sú spojené rovnakou farbou čiary. Údaje boli získané z tabuľky 6.3.

- zelená čiara - prahová hodnota RGB (200,200,200)
- čierna čiara - prahová hodnota RGB (169,169,169)
- sivá čiara - prahová hodnota RGB (150,150,150)
- žltá čiara - prahová hodnota RGB (130,130,130)
- modrá čiara - prahová hodnota RGB (100,100,100)

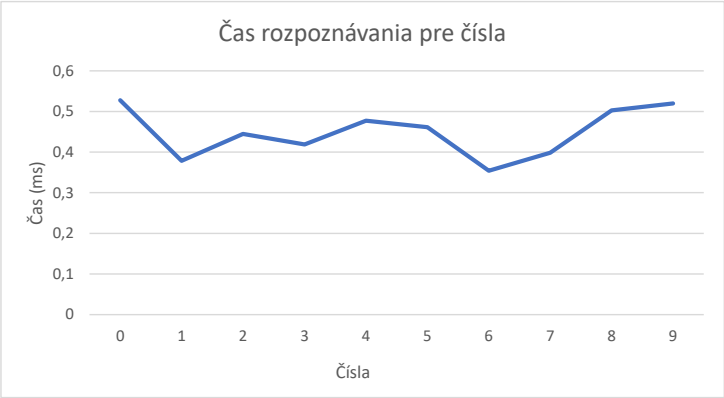
Pre niektoré čísla ako napríklad 2, alebo 6 zmena prahovej hodnoty nemá veľký dopad na úspešnosť. Stále sa držia na vysokých hodnotách. Naopak pre niektoré čísla ako 5, 8 alebo 9 táto zmena výrazne ovplyvňuje úspešnosť. Najlepšie výsledky boli dosiahnuté pri prahovej hodnote RGB (169,169,169). Priemer všetkých úspešností bol najvyšší.

Čísla	Prahová hodnota	Úspešnosť (%)				
		200	169	150	130	100
0		73	77,1242	70	70	71,2
1		78,5	81,4286	81,4	82,8	75,7
2		84,3	86,875	87,5	86,8	85
3		58,8	70,922	63	63,1	59,5
4		55,3	69,9029	66	60,1	58,2
5		55,8	73,5294	78	80,3	77,4
6		93	97,2028	96	95,8	95,1
7		60,9	74,2587	67	63,8	60,9
8		44	66,9643	54	51,7	42,8
9		53	88,6364	78	69,3	68,1
Priemer		65,66	78,68443	74,09	72,37	69,39

Obr. 6.3: Tabuľka prahových hodnôt

6.1.3 Rýchlosť rozpoznávania

číslo	čas (ms)
0	0,527623
1	0,378638
2	0,444531
3	0,4186
4	0,477562
5	0,461169
6	0,354138
7	0,398438
8	0,502838
9	0,519931
Priemer	0,448347



Obr. 6.4: Rýchlosť rozpoznávania

Testovanie rýchlosti rozpoznávania, ktoré možno vidieť na obrázku 6.4, bolo na číslach z upraveného datasetu MNIST. Pre každé číslo sa vybral náhodne jeden kandidát (obrázok). Následne sa ňom sa previedlo 15 meraní a výsledky sa zapísali do tabuľky 6.5. Obrázky, na ktorých boli čísla, mali veľkosť 300x300 pixelov. Výpočet prebiehal na stroji s procesorom Intel® Core™ i5-8300H. Čas zaznamenával iba proces rozpoznania. Čas načítania obrázka sa nepočítal.

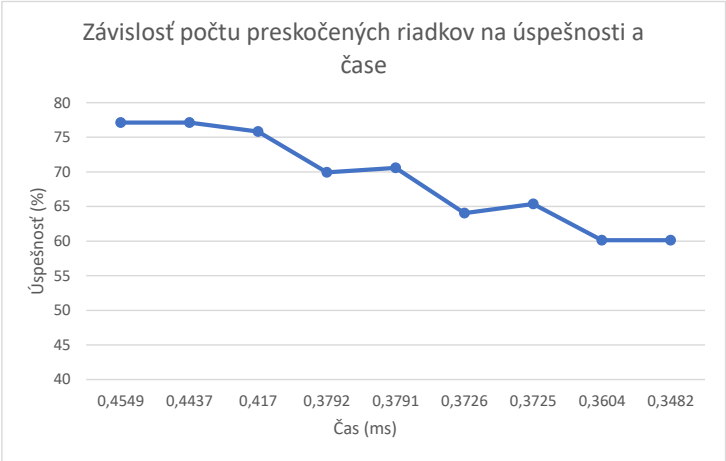
Hodnoty sa získali z tabuľky na obrázku 6.5. odstránili sa dve merania pre každé číslo (najkratší a najdlhší čas). Následne sa z nich vypočítal priemer, a ten sa vložil do grafu.

Číslo merania	Čas pre čísla (ms)									
	0	1	2	3	4	5	6	7	8	9
1	0,5564	0,4159	0,4599	0,3756	0,4846	0,4571	0,3497	0,4137	0,536	0,5549
2	0,496	0,3855	0,4521	0,4294	0,482	0,3374	0,4073	0,3909	0,5448	0,513
3	0,521	0,376	0,4603	0,3553	0,4733	0,5092	0,3339	0,4163	0,5396	0,5398
4	0,474	0,449	0,4775	0,2983	0,4744	0,4224	0,4246	0,3684	0,4578	0,5986
5	0,5725	0,4118	0,4824	0,4332	0,4383	0,4927	0,4072	0,3625	0,533	0,5132
6	0,5657	0,3768	0,3814	0,4803	0,4877	0,4042	0,3098	0,4573	0,431	0,489
7	0,5219	0,3829	0,3948	0,4321	0,4794	0,4861	0,4324	0,3341	0,5365	0,5571
8	0,5398	0,3084	0,461	0,4467	0,4991	0,4577	0,315	0,3816	0,3711	0,5103
9	0,4638	0,3791	0,4215	0,4329	0,4598	0,4608	0,331	0,4457	0,45	0,5386
10	0,5023	0,397	0,4297	0,4356	0,4874	0,5375	0,3389	0,4246	0,4729	0,4207
11	0,6277	0,3346	0,4864	0,4613	0,4785	0,4898	0,3197	0,3788	0,4795	0,5416
12	0,5563	0,3979	0,3502	0,4178	0,4698	0,4711	0,3697	0,376	0,5015	0,4449
13	0,5038	0,3364	0,4381	0,3462	0,49	0,4797	0,3469	0,421	0,5209	0,5287
14	0,4658	0,3986	0,4699	0,4408	0,4775	0,423	0,3819	0,4047	0,526	0,502
15	0,5856	0,356	0,4503	0,4349	0,4525	0,4414	0,2832	0,3955	0,5522	0,4845
Priemer	0,530173	0,380393	0,441033	0,414693	0,47562	0,458007	0,356747	0,398073	0,496853	0,515793

Obr. 6.5: Výsledky merania rýchlosti

6.1.4 Preskakovanie riadkov

Počet preskočených riadkov	Čas (ms)	Úspšnosť (%)
1	0,4549	77,1242
2	0,4437	77,1242
4	0,417	75,817
6	0,3792	69,9346
8	0,3791	70,5882
10	0,3726	64,0523
12	0,3725	65,3595
14	0,3604	60,1307
16	0,3482	60,1307



Obr. 6.6: Závislosť počtu preskočených riadkov na úspešnosti a čase

Každý bod v grafe na obrázku 6.6 reprezentuje počet preskočených riadkov. Na x-ovej ose je čas, ktorý sa znižuje, keď sa zväčšuje počet riadkov, ktoré sa preskočili. Tým sa ale môže ovplyvniť úspešnosť rozpoznania čísel. Tá tiež klesá.

Pre každé číslo z upraveného datasetu MNIST sa nameral čas. Do tabuľky sa zapísal priemer týchto časov.

Z grafu vyplýva, že sa neoplatí preskakovať príliš veľa riadkov. Zníži sa čas rozpoznania, ale úspešnosť klesne. Pri väčšom rozlíšení obrázkov bude tento efekt klesania úspešnosti menší a oplatí sa preskočiť aj viac riadkov.

6.2 Testovanie na vlastných číslach

Okrem testovania na už existujúcom datasete bola aplikácia testovaná ručne. Odfotili sa obrázky čísel na bielom podklade písané väčšinou s čiernou alebo modrou farbou pera.

Vznikol tak nový dataset, ktorý má pre každé číslo od 11-26 obrázkov. Dokopy sa tu nachádza 179 rôznych obrázkov čísel. Každý z nich má veľkosť (iný počet pixelov) približne v rozmedzí 100-200 pixelov na šírku a výšku.

Jedným z najväčších problém toho, že sa niektoré čísla nesprávne určili, bola transformácia farebného obrázku do vstupu pre automat, ktorý počíta iba z dvoma farbami.

Prahová hodnota pri tomto rozpoznávaní bola stanovená rovnaká ako pri predchádzajúcom datasete a to RGB(169,169,169)

Číslo	Úspešnosť %	Počet čísel	0	1	2	3	4	5	6	7	8	9
0	81,82%	11	9	0	1	0	0	0	1	0	0	0
1	95,83%	24	0	23	0	0	0	1	0	0	0	0
2	80,77%	26	0	5	21	0	0	0	0	0	0	0
3	80,00%	15	0	2	0	12	0	1	0	0	0	0
4	87,50%	16	0	0	0	0	14	0	0	0	0	0
5	65,00%	20	0	0	0	0	0	13	4	0	0	0
6	88,89%	18	0	0	0	0	0	1	16	0	0	0
7	62,50%	16	0	0	0	2	0	2	0	10	0	0
8	73,33%	15	0	0	0	0	0	3	0	0	11	0
9	77,78%	18	0	1	0	0	0	1	2	0	0	14
Celkový počet čísel		179										
Priemer	79,34%											

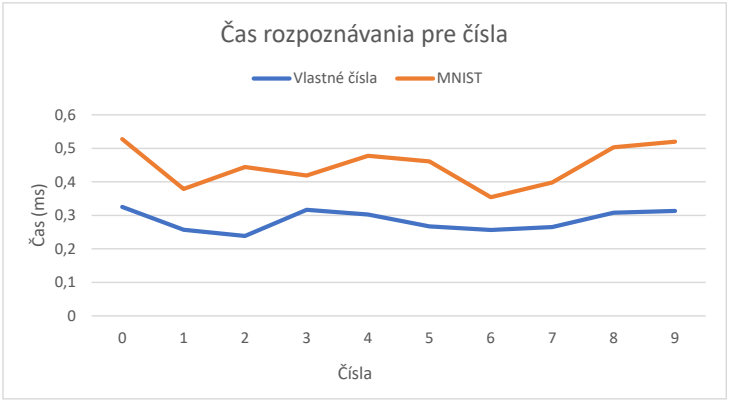
Obr. 6.7: Tabuľka výsledkov pri vlastných číslach

Výsledky sú zobrazené na obrázku 6.7 v tabuľke. Prvý stĺpec reprezentuje testované číslo. Ďalší stĺpec zobrazuje úspešnosť rozpoznania jednotlivého čísla. Stĺpec s názvom počet čísel obsahuje hodnotu, na koľkých číslach sa rozpoznávalo. Ostatné stĺpce ukazujú koľkokrát sa našlo ktoré číslo.

Priemerná úspešnosť jednotlivých čísel je 79.34 %.

Rýchlosť rozpoznávania

číslo	čas (ms)
0	0,3252
1	0,2571
2	0,2385
3	0,3164
4	0,3026
5	0,2671
6	0,2567
7	0,2649
8	0,3081
9	0,3135
Priemer	0,28501



Obr. 6.8: Rýchlosť rozpoznávania pri vlastných číslach

Testovanie času rozpoznávania bolo merané na datasete vlastných čísel. Odmeral sa čas pre každé číslo, výsledky sa spriemerovali a vložili do tabuľky 6.8.

Na grafe vyššie sú zobrazené časy ako pre vlastné čísla, tak aj pre upravený dataset MINIST. V oboch prípadoch je priebeh približne rovnaký. Pri vlastných číslach majú obrázky menšie rozlíšenie, tak aj čas je menší.

Kapitola 7

Záver

Cieľom tejto práce bolo zaviesť dvojdimenzionálnu verziu skákajúcich automatov, implementovať automat a následne demonštrovať na aplikácii. Výsledná aplikácia umožňuje rozpoznávanie čísel z obrázka. Tá bola vytvorená spojením dvoch typov, dvojrozmerného konečného automatu a skákajúceho automatu.

Na začiatku práce boli v skratke zhrnuté a opísané rôzne varianty klasických konečných automatov. Či už sa pohybovali po páske jedným smerom, alebo oboma. Ku každému automatu bol uvedený aj konkrétny príklad, kvôli lepšiemu pochopeniu a oboznámeniu sa s ním.

V ďalšej sekcii boli spomenuté skákajúce automaty, ktoré majú špecifický pohyb. Namiesto sekvenčného pohybu, čítacia hlavica skáče medzi bunkami vstupnej pásky. Dávajú tak nový pohľad a spôsob pohybu po vstupe. Pokračovalo sa automatmi, ktoré už nemali vstupnú pásku jednorozmernú. Vo väčšine prípadov sa jednalo o dvojrozmernú, ktorá reprezentovala vstupný obrázok. Opísali sa tu ich rôzne variácie ako Štvorcestné automaty, alebo On-line Teselačné.

Nasledovala kapitola, ktorá prinášala nový typ automatu na rozpoznávanie útvarov z obrázka. Bol tvorený spojením dvoch typov, dvojrozmerného konečného automatu a skákajúceho automatu. Umožňoval chodiť po vstupnej páske do všetkých smerov (hore, dole, doprava, doľava) podobne ako štvorcestný automat. Zároveň však má definované aj skoky. Tie mu umožňovali ľubovoľne skákať po vstupe. Bolo veľa spôsobov, ako tieto vlastnosti využiť. Napríklad sa mohli rozpoznávať objekty klasickým spôsobom a po skončení rozpoznávania sa v momente premiestniť na opačnú stranu vstupnej pásky a pokračovať ďalej.

Na rozpoznávanie čísel bolo využitých viacero menších automatov, ktoré umožňovali rozoznať niektoré základe útvary ako horizontálna, vertikálna, alebo šikmá čiara. Vyzerali podobne, avšak rozdiel v nich bol v tom, do ktorého primárneho smeru sa pohybovali. To, aký typ čiary sa rozpoznal, záviselo okrem iného aj na tom, koľkokrát sa vošlo do určitého stavu. Ak sa napríklad automat po väčšinu času nachádzal v stave, ktorý reprezentoval pohyb smerom dole, tak bola veľká šanca, že práve rozpoznaný útvar (čiara) bude vertikálna. Jednotlivé automaty sa spájali, aby práca s nimi mohla byť jednoduchšia.

To, aký automat sa spustil, poprípade či sa udiala nejaká iná udalosť, rozhodoval rozhodovací strom. V každom uzle mal informáciu, ako bude rozpoznávanie ďalej prebiehať. Začínalo sa v koreňovom uzly a pohybovalo sa smerom dole ku listovým. V listových uzloch rozpoznávanie končilo.

Využitím tohto typu automatu sa v ďalšej časti realizovala ich aplikácia. Je veľa spôsobov ako ich použiť. Táto práca sa zamerala na rozpoznávanie čísel. Program dostal na vstup obrázok, kde mohol byť ľubovolný počet čísel a po vyhodnotení sa zobrazia všetky

nájdené čísla. Využitie takejto aplikácie by mohlo byť napríklad v automatickom nájdení a vyhodnocovaní matematických výrazov v obrázku.

Existuje veľa spôsobov ako sa môžu čísla rozpoznávať z obrázka. Niektoré sú lepšie, iné sú horšie. Jedna z najlepších metód je rozpoznávanie pomocou neurónových sietí. Tie dosahujú vysokú mieru presnosti, ak sú správne natrénované. Majú však nevýhodu, že proces trénovania trvá dlhší čas. Dôležitý je aj dataset, na ktorom sa učia. Musí byť dostatočne veľký a rozmanitý. Pri správnom natrénovaní dokážu rozoznať aj čísla, ktoré nie sú celé spojité.

V tejto práci bol vyskúšaný iný pohľad na danú problematiku. Výhoda oproti napríklad neurónovým sieťam je ten, že priebeh rozpoznávania trvá rýchlejšie. Nie je ani potrebné mať na spustenie výkonný software. Neprebiehajú tu zložité výpočty. Čítacia hlavica sa posúva po páske, poprípade zmení svoju polohu. Má aj svoje nevýhody. Napríklad to, že je potrebné správne vytvoriť automaty tak, aby pracovali korektne, čo nie je vždy najľahšia práca. Pre každý pohyb musia byť definované pravidlá.

V testovaní bolo ukázané ako pracuje daná implementácia. Testovalo sa na dvoch datasetoch. Jeden bol upravený dataset MNIST, ktorý sa používa pre trénovanie a testovanie neurónových sieti. Ďalší bol vytvorený fotením obrázkov čísel na bielom pozadí. Čísla mali rôznu veľkosť. Bola vytvorená tabuľka s úspešnosťou pre jednotlivé čísla, ktoré mali úspešnosť okolo 80%. Neurónové siete dosahujú oveľa väčšiu presnosť až 98%.

Testovala sa aj rýchlosť nájdenia čísla v obrázku. Nameralo sa 15 hodnôt pre každé číslo. Odstránili sa dva časy (najkratší a najdlhší čas) a výsledné hodnoty sa spriemerovali. Vznikol graf, ktorý znázorňoval časy pre jednotlivé čísla. Niektoré časy jemne vyčnievali či už nahor alebo nadol. Najviac času zaberalo samotné načítanie čísla. Vyhodnotenie bolo už rýchle.

Taktiež tu boli skúšané rôzne optimalizácie, napríklad preskakovanie riadkov z dôvodu ušetrenia času. Pri pár preskočených riadkoch bol čas aj úspešnosť nájdenia približne rovnaký ako bez preskakovania. Veľká zmena nastala pri skákaní cez viac riadkov. Úspešnosť začala výrazne klesať, čas sa už veľmi nemenil. Neoplatilo sa preskakovať príliš veľa riadkov. Pri väčšom rozlíšení obrázka by bol tento efekt klesania menší.

Najväčší vplyv na úspešné rozpoznanie mal prevod obrázka do reprezentácie, s ktorou by vedel automat pracovať. Ak mal obrázok väčšie rozlíšenie, prevod bol lepší a tým pádom aj samotné nájdenie výsledku malo tak vyššiu úspešnosť. Do budúcna by sa vylepšenie mohlo zamerať práve na tento prevod.

Poprípade by sa mohol použiť automat, ktorý by v jednom kroku čítal naraz viac buniek vstupnej pásky. Napríklad by sa čítali štyri, alebo deväť buniek vedľa seba. Mohlo by to pomôcť k výrazne lepšej úspešnosti v prípade rôznych nepresností v čísle. S tým ale prichádzajú ďalšie problémy ohľadom rozlíšenia. Vstupný obrázok by musel byť dostatočne veľký a takisto aj rozpoznávané číslo by nemalo byť pár pixelov široké a vysoké.

Literatúra

[1] Carroll, J. a Long, D. *Theory of Finite Automata With an Introduction to Formal Languages*. Prentice Hall, 1989. ISBN 9780139137082.

[2] Hopcroft, J. E., Motwani, R. a Ullman, J. D. *Introduction to Automata Theory, Languages, and Computation*. 2. vyd. Addison Wesley, 2000. ISBN 0-201-44124-1.

[3] Kozen, D. C. *Automata And Computability*. Springer Science & Business Media, 2012. ISBN 978-1-4612-1844-9.

[4] LeCun, Y. a Cortes, C. *THE MNIST DATABASE of handwritten digits* [online]. [cit. 2022-04-18]. Dostupné z: http://yann.lecun.com/exdb/mnist/.

[5] Meduna, A. a Soukup, O. *Modern Language Models and Computation: Theory with Applications*. Springer, 2017. ISBN 978-3-319-63099-1.

[6] Mehta, D. P. a Sahni, S. *Handbook Of Data Structures And Applications*. 2. vyd. Chapman and Hall/CRC, 2018. ISBN 9781498701853.

[7] Rozenberg, G. a Salomaa, A. *Handbook of Formal Languages: Volume 3 Beyond Words*. Springer, 2012. ISBN 9783642638596.

[8] The Qt Company. *Cross-platform software development for embedded & desktop* [online]. [cit. 2022-04-18]. Dostupné z: https://www.qt.io/product/.

Príloha A

Obsah CD

- **src/** - adresár so zdrojovými kódmi aplikácie
- **modify_MNIST/** - adresár s obrázkami čísel upraveného datasetu MNIST
- **custom_Dataset/** - adresár s obrázkami čísel vlastného datasetu
- **apk/** - adresár so spustiteľnou aplikáciou
- **xsvacd00.pdf** - text diplomovej práce
- **doc/** - adresár so zdrojovými kódmi textu diplomovej práce
- **README.md** - stručný manuál ku spusteniu