Vizualizace multimnožinových konečných automatů

Miloš Petrenčák

Bakalářská práce
2020

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Ústav automatizace a řídicí techniky

Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Miloš Petrenčák**
Osobní číslo:	**A17721**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**Prezenční**
Téma práce:	**Vizualizace multimnožinových konečných automatů**
Téma práce anglicky:	**The Visualisation of Multiset Finite Automata**

Zásady pro vypracování

1. Popište teorii týkající se multimnožinových konečných automatů (dále MKA).
2. Do popisu zahrňte MKA bez detekce i s detekcí, deterministické i nedeterministické.
3. Ve vhodném programovacím jazyce sestavte simulátor různých variant MKA.
4. Simulátor musí umožňovat jednoduché zadávání MKA ve zvolené variantě, jejich editaci a názornou vizualizaci jejich činnosti.

Rozsah bakalářské práce:
Rozsah příloh:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. E. Csuhaj-Varjú, C. Martín-Vide a V. Mitrana, „Multiset automata" v Multiset processing – mathematical, computer science, and molecular computing points of view, C. S. Calude, G. Paun, G. Rozenberg a A. Salomaa, Ed., Lecture notes in computer science, díl 2235, Berlín: Springer, 2001, str. 69-83.
2. Černá, Ivana, a kol. Automaty a formální jazyky I. Učební text FI MU. Fakulta informatiky, Masarykova univerzita, Brno: 2002. Dostupné z: http://is.muni.cz/elportal/estud/fi/js06/ib005/Formalni_jazyky_a_automaty_I.pdf
3. M. Kudlek, C. Martín-Vide, and G. Paun, „Toward a formal macroset theory" v Multiset processing – mathematical, computer science, and molecular computing points of view, C. S. Calude, G. Paun, G. Rozenberg a A. Salomaa, Ed., Lecture notes in computer science, díl 2235, Berlín: Springer, 2001, str. 123-133.
4. M. Kudlek, P. Totzke a G. Zetsche, „Multiset pushdown automata", Fundamenta Informaticae, díl 93, str. 221-233, 2009.
5. M. Sipser, Introduction to the Theory of Computation, 2. vyd., Boston: Thomson Course Technology, 2006.

Vedoucí bakalářské práce: **Ing. Pavel Martinek, Ph.D.**
Ústav matematiky

Datum zadání bakalářské práce: **20. prosince 2019**
Termín odevzdání bakalářské práce: **15. května 2020**

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

Ve Zlíně dne 20. prosince 2019

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 12.08.2020

Miloš Petrenčák
podpis diplomanta

ABSTRAKT

Předmětem této práce je vývoj aplikace pro vizualizaci výpočtu multimnožinového konečného automatu. Práce je rozčleněna na část teoretickou a praktickou. V teoretické části je popsána problematika multimnožinového konečného automatu a dalších základních pojmů. Praktická část je zaměřena na vývoj aplikace. Součástí aplikace je i grafické uživatelské rozhraní, které uživateli umožňuje zadávání různých variant a následnou vizualizaci výpočtu krok po kroku. Aplikace dále umožňuje ukládání a načítání vstupních parametrů na pevný disk. Cílem aplikace je přehledná vizualizace výpočtu, která může být využita např. k výuce.

Klíčová slova: vizualizace, multimnožina, konečný automat, grafické uživatelské rozhraní

ABSTRACT

The main gold of this thesis is development of an application for visualization of the calculation of a multiset finite automata. The work is divided into theoretical and practical part. The teoretical part describes the issue of multiset finite automata and other basic concepts. The practical part is focused on application development. The application also includes a graphical user interface, which allows the user to enter various variants and then visualize the calculation step by step. The application also allows saving and loading input parameters to the hard disk. The aim of the application is a clear visualization of the calculation, which can be used, for example, for teaching.

Keywords: visualization, multiset, finite automata, graphical user interface

Především bych chtěl poděkovat vedoucímu své bakalářské práce, panu Ing. Pavlu Martinkovi Ph.D, za trpělivost, ochotu a především za cenné rady při vypracovávání práce. Dále bych chtěl celé své rodině za jejich podporu po celou dobu mého studia.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Předmětem této bakalářské práce je popis různých variant multimnožinových konečných automatů a vývoj desktopové aplikace pro vizualizaci jejich výpočtů. V práci se zabývám deterministickými a nedeterministickými multimnožinovými konečnými automaty s detekcí, a i bez detekce.

Teoretická část práce se zabývá formálním popisem všech zmíněných variant multimnožinových konečných automatů, také jsou zde charakterizovány všechny pojmy, které jsou k tomuto popisu nutné. Jelikož k tématu multimnožinových konečných automatů nejsou české materiály, bude se jednat o jedny z prvních českých materiálů na toto téma.

Praktická část práce se zabývá vývojem desktopové aplikace. Hlavní ideou při návrhu aplikace byla jednoduchost a přehlednost. Aplikace jsem pojmenoval *Simulátor multimnožinových konečných automatů*. Tato aplikace je primárně určena pro operační systém Windows. Pro vývoj aplikace je využito vývojové prostředí *Visual Studio 2019* a programovací jazyk je C#.

Mou hlavní motivací pro volbu tohoto tématu bylo zlepšení se ve vývoji softwaru.

I. TEORETICKÁ ČÁST

1 ZÁKLADNÍ POJMY

1.1 Multimnožina

Multimnožina je zobecněním pojmu množina. Základní rozdíl mezi množinou a multimnožinou je takový, že multimnožina umožňuje vícenásobný výskyt jejich prvků.

Multimnožinou nad množinou Σ rozumíme zobrazení $\alpha: \Sigma \rightarrow \mathbb{N}$, kde $\mathbb{N}$ označuje množinu všech přirozených čísel včetně 0. [1; 2]

Abecedou nazýváme libovolnou, konečnou, neprázdnou množinou. Množina všech multimnožin nad abecedou Σ označujeme jako $\Sigma^{\oplus}$. [1; 2]

Normu $|\alpha|$ multimnožiny α definujeme:

$$|\alpha| = \sum_{x \in \Sigma} \alpha(x), \tag{1}$$

kde $\alpha(x)$ označuje četnost výskytu prvku x v multimnožině. [1]

Prázdnou multimnožinu označujeme $\mathbf{0}_{\Sigma}$. Myslíme tím multimnožinu splňující následující podmínku z rovnice (2) [1; 2]

$$\mathbf{0}_{\Sigma}(x) = 0, x \in \Sigma \tag{2}$$

Multimnožinovým rozdílem $\ominus$ rozumíme operaci nad multimnožinami definovanou následovně. Jsou-li α, β multimnožiny, pak jejich rozdílem rozumíme multimnožinu, která splňuje následující podmínky:

$$\alpha(x) \ominus \beta(x) = max\{0, \alpha(x) - \beta(x)\}, x \in \Sigma \tag{3}$$

Příklad: {1, 1, 2} $\ominus$ {2, 2, 3} = {1, 1}. Jako další příklad můžeme uvést {2, 2, 3}$\ominus${1, 1, 2} = {2, 3}.

Multimnožina se dá například použít k popisu rozkladu přirozeného čísla na prvočinitele, konkrétně číslo 120 můžeme reprezentovat multimnožinou {2,2,2,3,5}.

Pro jednoprvkovou multimnožinu budeme používat označení $\langle y \rangle$. Tím označením budeme rozumět multimnožinu, pro niž platí následující rovnice (4). [1; 2]

$$\langle y \rangle(x) = 0 \; pro \; x \neq y \; a \; \langle y \rangle(y) = 1 \tag{4}$$

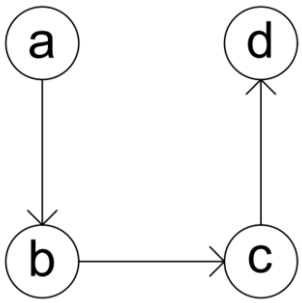
1.2 Relace, uzávěry relací

Relace udává vztah mezi objekty. V našem případě budeme nejčastěji používat relaci mezi dvěma objekty. Takovou relaci nazýváme binární. Binární relace zapisujeme pomocí uspořádaných dvojic a můžeme je znázorňovat pomoci orientovaných grafů. Tyto orientované grafy se skládají z množiny uzlů, z nichž některé dvojice jsou spojeny pomoci orientovaných hran. [3]

Máme-li například relaci, uvedenou jako rovnici (5),

$$R = \{(a, b), (b, c), (c, d)\} \quad (5)$$

na množině {a, b, c}. Tuto relaci můžeme znázornit pomocí orientovaného grafu, jenž je uveden na *obrázku 2*.

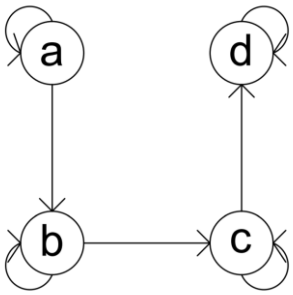


Obrázek 1: relace R

Reflexivním uzávěrem relace R na množině X je relace R', která splňuje následující vlastnosti:

- Pro všechna $x \in X$ platí, že uspořádaná dvojice $(x, x) \in R'$.
- Pro všechna $x, y \in X, x \neq y$ platí, že uspořádaná dvojice $(x, y) \in R'$ právě, když $(x, y) \in R$. [3]

Uvažme relaci R zobrazenou na *obrázku 1.* Reflexivní uzávěr R' relace R máme zobrazen na *obrázku 2*.

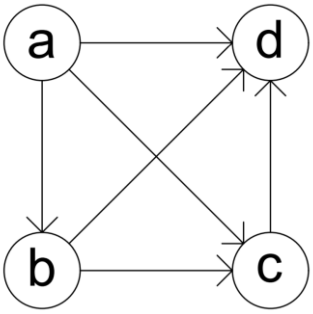


Obrázek 2: reflexivní uzávěr relace R

Tranzitivním uzávěrem relace R na množině X je relace R'' definovaná induktivně:

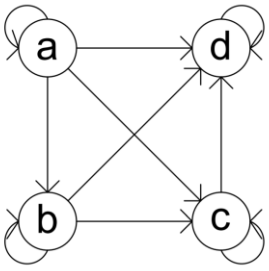
- Pro všechna $x, y \in X, (x, y) \in R$, platí $(x, y) \in R''$.
- Pro všechna $x, y, z \in X, (x, y) \in R'', (y, z) \in R''$, platí $(x, z) \in R''$. [3]

Uvažme relaci R zobrazenou na *obrázku 1*. Tranzitivní uzávěr R'' relace R máme zobrazen na *obrázku 3*.



Obrázek 3: tranzitivní uzávěr relace R

Reflexivním a tranzitivním uzávěrem $\tilde{R}$ nazýváme sjednocení reflexivního uzávěru R' *(obrázek 2)* a tranzitivního uzávěru R'' *(obrázek 3)*. [3] Reflexivní a tranzitivní uzávěr $\tilde{R}$ máme tedy znázorněn na *obrázku 4*.



Obrázek 4: reflexivní a tranzitivní uzávěr relace R

1.3 Konečné automaty

Pro lepší pochopení multimnožinových konečných automatů si v této kapitole představíme řetězcové konečné automaty.

Nejjednodušším automatem je konečný automat. Tento automat nemá žádné speciální paměťové zařízení. Multimnožinový konečný automat je stroj, který se musí nacházet v některém z konečně mnoha stavů. Automat na základě vstupního podnětu přejde do jiného stavu, nebo setrvá v aktuálním stavu. Dvojici aktuální stav a vnější podnět odpovídá vždy stejný výsledný stav. [3; 4]

Pro konkrétní představu si můžeme multimnožinový konečný automat představit jako zařízení skládající se:

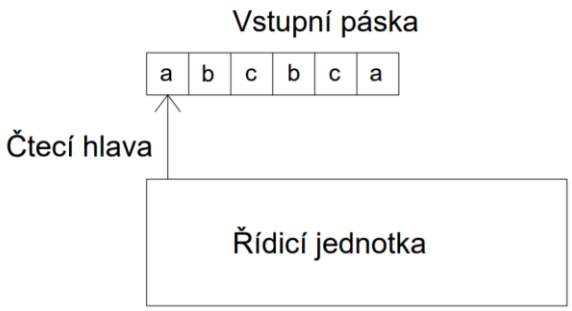
- z řídicí jednotky,
- ze vstupní pásky,
- ze čtecího zařízení.

Řídící jednotka se může nacházet v některém z konečně mnoha stavů.

Na vstupní pásce je zapsán řetězec symbolů představujících posloupnost podnětů, jejichž působení bude konečný automat vystaven.

Čtecí zařízení slouží také ke čtení jednotlivých symbolů z pásky. [3; 4]

Pro lepší představu je konečný automat ilustrován na *obrázku č. 5*.



Obrázek 5: Ilustrace konečného automatu

1.3.1 Popis činnosti konečného automatu, přechodová funkce

Na začátku své činnosti se konečný automat nachází ve vyznačeném tzv. počátečním stavu. Na *obrázku č. 5* je tento stav znázorněn tím, že je čtecí hlava umístěna na prvním (tj. nejlevějším) políčku čtecí pásky. Poté automat přečte první symbol na vstupní pásce. Na základě přečteného vstupního symbolu a aktuálního stavu přejde do stavu nového, nebo setrvá v aktuálním stavu. [3]

Informace o tom, jak se má konečný automat konkrétně zachovat, musí být součástí popisu každého konečného automatu. Tato informace bude popsána pomocí přechodové funkce konečného automatu. Přechodová funkce přiřazuje každé dvojici (stav, vstupní symbol) konkrétní stav. [3]

Přechodovou funkci můžeme graficky znázornit pomocí tzv. stavového diagramu. Stavové diagramy jsou založeny na orientovaných grafech, kde uzly představují stavy automatu a hrany představují přechody automatu mezi jednotlivými stavy. Pokud je nějaký uzel znázorněn dvojitě, odpovídající stav je koncový. Pokud orientovaná šipka nezačíná v žádném uzlu potom stav, do kterého tato šipka míří nazýváme počáteční stav. [3]

Automat na základě aktuálního stavu a aktuálně čteného symbolu vypočte následující stav. Jakmile automat přečte všechny symboly na vstupní pásce skončí zřejmě v nějakém stavu. Pokud půjde o nějaký ze stavů, jež budeme nazývat koncových, pak budeme tento výsledek chápat tak, že konečný automat příslušné vstupní slovo přijal (akceptoval). Pokud konečný automat skončí svou činnost v některém ze stavů, které nejsou koncové, budeme to chápat jako zamítnutí tohoto vstupního slova. [3]

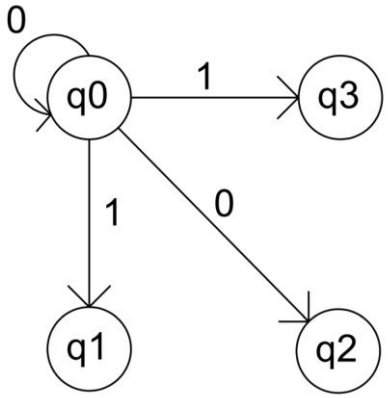
1.3.2 Deterministický konečný automat

Konečný automat nazýváme deterministický, pokud stav do kterého má automat přejít je jednoznačně určen aktuálním stavem a čteným symbolem. [3]

1.3.3 Nedeterministický konečný automat

Jistým zobecněním deterministických konečných automatů jsou nedeterministické konečné automaty. U nedeterministických konečných automatů nepožadujeme, aby následující stav byl jednoznačně určen aktuálním stavem a čteným symbolem. Připouštíme existenci celé (případně i prázdné) množiny stavů, z nichž může automat v příslušném výpočetním kroku kterýkoliv vybrat. [3]

Pro lepší představu uvažme nedeterministický konečný automat znázorněný na *obrázku 6*.



Obrázek 6: Nedeterministický konečný automat

Na *obrázku 6* vidíme nedeterministický konečný automat znázorněný pomocí stavového diagramu. Tento nedeterministický konečný automat má stavy q0, q1, q2 a q3. Pokud se automat nachází ve stavu q0 vidíme, že při přečtení symbolu 1 může automat přejít do stavu q1 nebo do stavu q3.

2 MULTIMNOŽINOVÝ KONEČNÝ AUTOMAT

Multimnožinový konečný automat pracuje na stejném principu jako konečný automat pracující s řetězci. Jediným rozdílem mezi multimnožinovým a řetězcovým automatem je takový, že multimnožinový konečný automat pracuje na vstupu s libovolnou multimnožinu.

Multimnožinový nedeterministický konečný automat je formálně definován jako uspořádaná pětice.

$$M = (\mathrm{Q}, \Sigma, \delta, \mathrm{q}_0, \mathrm{Q}_F), kde \tag{6}$$

- Q je neprázdná konečná množina stavů.
- Σ je konečná množina vstupních symbolů (tzv. vstupní abeceda).
- $\delta : \mathrm{Q} \times \Sigma \rightarrow 2^Q$ je přechodová funkce, kde 2^Q je množina všech podmnožin množiny Q.
- $\mathrm{q}_0 \in \mathrm{Q}$ je počáteční stav.
- $\mathrm{Q}_F \subseteq \mathrm{Q}$ je množina koncových stavů.

Konfigurace multimnožinového konečného automatu je uspořádaná dvojice

$$(q, \mu) \in \mathrm{Q} \times \Sigma^{\oplus}, \tag{7}$$

kde q je aktuální stav a μ je dosud nepřečtená multimnožina. Jinými slovy, konfigurace multimnožinového konečného automatu tedy znamená, v jakém stavu se multimnožinový konečný automat nachází a jaké symboly zbývají z multimnožiny přečíst. [1; 5]

Na množině konfigurací definujeme následující binární relaci reprezentující výpočetní krok.

$$(q, \mu) \vdash (q', \mu'), \tag{8}$$

pokud existuje $a \in \Sigma$ takové, že četnost výskytu $\mu(a) \geq 1$,

$$q' \in \delta(\mathrm{q}, \mathrm{a}) \tag{9}$$

a

$$\mu \ominus \langle a \rangle = \mu' \tag{10}$$

Po úspěšném přečtení vstupního symbolu jej multimnožinový konečný automat odstraní z vstupní multimnožiny. Multimnožinový konečný automat přijímá multimnožinu μ, pokud na vstupu po konečně mnoho krocích nezůstane žádný prvek a automat se nachází

v koncovém stavu. Jazyk M(A) akceptovaný multimnožinovým konečným automatem se skládá ze všech přijímaných multimnožin.

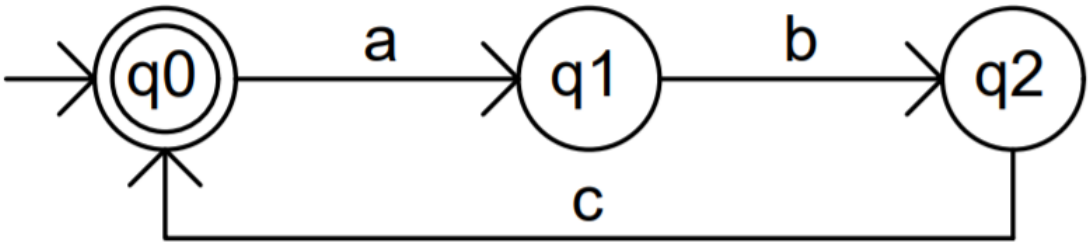
$$M(A) = \{\mu \in \Sigma^{\oplus} \mid \exists q \in Q_F : (q_0, \mu) \vdash^* (q, \mathbf{0}_\Sigma)\}, \tag{11}$$

kde $\vdash^*$ je reflexivní a tranzitivní uzávěr relace $\vdash$. [1; 5]

Jako příklad multimnožinového konečného automatu si můžeme představit následující automat

$$A = (\{q_0, q_1, q_0\}, \{a, b, c\}, \delta, \{q_0\}, q_0), \tag{12}$$

jehož přechodová funkce δ je vyobrazena na *obrázku 7*.



Obrázek 7: Přechodová funcke automatu A [5]

Ze zadání automatu A můžeme vyčíst, že stavy automatu jsou q_0, q_1, q_2, přičemž q_0 označuje jak počáteční, tak koncový stav. Množinu všech vstupních symbolů tvoří symboly a, b, c.

Nastíníme si tedy průběh výpočtu takového automatu. Můžeme sestavit multimnožinu $m_1 = \{a, a, b, c\}$. Z přechodové funkce je zřejmé, že jediná možnost přechodu ze stavu q_0 do stavu q_1 je taková, že automat přečte na vstupu symbol *„a“*. Po úspěšném přečtení, automat symbol *„a“* odstraní v multimnožině m_1. Zůstávají tedy symboly {a, b, c}. Nyní se automat nachází ve stavu q_1. Ve stavu q_1 vidíme, že další přechod nastane tehdy, pokud automat přečte ze vstupní multimnožiny symbol *„b“*. Tento symbol automat opět odstraní z multimnožiny m_1 a přejde do stavu q_2. Na vstupu automatu tedy zbývá {a, c}. Automat se nachází ve stavu q_2. Automat přečte ze vstupu symbol *„c“*, odstraní jej z multimnožiny m_1 a přejde do stavu q_0. Poté už jen automatu zbývá přečíst poslední symbol ze vstupní multimnožiny. Po přečtení symbolu *„a“* přejde MKA ze stavu q_0 do stavu q_1. Posléze už je multimnožina m_1 prázdná a automat se nachází ve stavu q_1, což není koncový stav. Multimnožinový konečný automat vstupní multimnožinu tedy nepřijímá.

Jako druhou multimnožinu uvažme $m_2 = \{a, b, c\}$. Automat se tedy nachází v počátečním stavu q_0. Ve stavu q_0 automat přečte symbol „a“, přejde do stavu q_1 a vyřadí symbol *„a“* z multimnožiny m_2. Poté automat ve stavu q_1 přečte symbol *„b“*, přejde do stavu q_2 a symbol *„b“* odstraní z multimnožiny m_2. Ve stavu q_2 přečte automat poslední symbol z multimnožiny m_2. Automat symbol *„c“* po přečtení odstraní z multimnožiny m_2 a přejde do koncového stavu q_0. Multimnožinový konečný automat již nemá na vstupu žádné symboly a nachází se v koncovém stavu. Multimnožinový konečný automat tedy přijímá vstupní multimnožinu {a, b, c}.

Z *obrázku 7* můžeme snadno postřehnout, že automat s takto zadanou přechodovou funkcí přijímá multimnožiny, které obsahují stejný počet prvků a, b, c. Můžeme tedy psát:

$$M\ (A) = \{\mu\ \in\ \{a, b, c\}^{\oplus} \mid \mu(a) =\ \mu(b) =\ \mu(c)\}.\ [5] \qquad (13)$$

2.1 Deterministický multimnožinový konečný automat

Multimnožinový deterministický konečný automat je formálně definován jako pětice, kde

$$M = (\mathrm{Q}, \Sigma, \delta, \mathrm{q}_0, \mathrm{Q}_F), \text{kde} \tag{14}$$

- Q je neprázdná konečná množina všech stavů.
- Σ je konečná množina vstupních symbolů (tzv. vstupní abeceda).
- $\delta : \mathrm{Q} \times \Sigma^{\oplus} \rightarrow Q$ je přechodová funkce.
- $\mathrm{q}_0 \in \mathrm{Q}$ je počáteční stav.
- $\mathrm{Q}_F \subseteq \mathrm{Q}$ je množina koncových stavů.

Tento automat nazýváme deterministickým, pokud jsou splněny následující podmínky:

- $|\delta(q, a)| \leq 1$ pro všechna $q \in \mathrm{Q}$ a $a \in \Sigma$.
- Pro každý stav q, když $|\delta(q, a)| \neq 0$, potom $|\delta(q, b)| = 0$ pro všechna $b \neq a$.

Stejně jako u konečných automatů, které pracují nad řetězci, se i zde nabízí otázka vztahu mezi deterministickým a nedeterministickým multimnožinovým konečným automatem. U konečných automatů, které pracují nad řetězci, můžeme deterministické a nedeterministické konečné automat navzájem mezi sebou převádět. Konečné automaty, které pracují nad řetězci, jsou tedy výpočetně stejně silné. U multimnožinových konečných automatů toto tvrzení neplatí. Větší výpočetní sílou disponují multimnožinové nedeterministické konečné automaty. [5]

3 MULTIMNOŽINOVÝ KONEČNÝ AUTOMAT S DETEKCÍ

Definujme nyní multimnožinový konečný automat s nepatrným vylepšením. Tento automat budeme nazývat multimnožinový konečný automat s detekcí (MKAD). Multimnožinový konečný automat s detekcí je schopen přejít ze stavu do stavu, nejen při čtení symbolu ze vstupní multimnožiny, ale také když zjistí, že některé symboly se ve vstupní multimnožině nevyskytují. Formální definice tohoto automatu je:

$$M = (\mathrm{Q}, \Sigma, \delta, \mathrm{Q}_F, \mathrm{q}_0,), kde \tag{15}$$

- Q je neprázdná konečná množina stavů.
- Σ je konečná množina vstupních symbolů (tzv. vstupní abeceda).
- $\delta : \mathrm{Q} \times (\Sigma \cup \bar{\Sigma}) \rightarrow 2^Q$ je přechodová funkcí, kde 2^Q je množina všech podmnožin množiny Q.
- $\mathrm{Q}_F \subseteq \mathrm{Q}$ je množina koncových stavů.
- $\mathrm{q}_0 \in \mathrm{Q}$ je počáteční stav.

Označením $\bar{\Sigma}$ máme na mysli množinu duálních symbolů k symbolům vstupní abecedy Σ. Tyto duální symboly budou reprezentovat chybějící symboly v aktuální multimnožině na vstupu. [5] V simulačním programu, z důvodu snadnější implementace, jsou tyto duální symboly zobrazovány velkými písmeny.

Výpočetní krok, který je definován pomocí přechodové funkce δ, která je aplikovaná na dvojici (q, a), kde $a \in \Sigma$ je stejný jako výpočetní krok, který je zavedený u multimnožinového konečného automatu. U multimnožinového konečného automatu s detekcí dále definujeme výpočetní krok popsaný přechodovou funkcí δ, která je aplikovaná na dvojici $(q, \bar{a})$, kde $\bar{a} \in \bar{\Sigma}$ určuje přechod do dalšího stavu z množiny $\delta(q, \bar{a})$. Tento výpočetní krok odpovídá situaci, že se symbol *„a“* nevyskytuje v multimnožině. Přesněji:

$$(q, \mu) \vdash (s, \rho), \tag{16}$$

pokud je splněná některá z následujících podmínek,

- existuje $a \in \Sigma$ takové, že $s \in \delta(q, a)$, $\mu(a) \geq 1$ a $\mu \ominus \langle a \rangle = \rho$,
- existuje $\bar{a} \in \bar{\Sigma}$ takové, že $s \in \delta(q, \bar{a})$, $\mu(a) = 0$, $\rho = \mu$.

Tohle vylepšení multimnožinového nedeterministického konečného automatu nepřinese žádné navýšení výpočetní síly. Ovšem pro deterministický multimnožinový konečný automat tohle vylepšení přidává další možnosti výpočtů. [5]

3.1 Deterministický multimnožinový konečný automat s detekcí

Deterministický multimnožinový konečný automat s detekcí je speciálním případem deterministického multimnožinového automatu.

Formálně definujeme deterministický multimnožinový automat s detekcí jako uspořádanou pětici:

$$M = (\mathrm{Q}, \Sigma, \delta, \mathrm{Q}_F, \mathrm{q}_0,), kde \tag{17}$$

- Q je neprázdná konečná množina stavů.
- Σ je konečná množina vstupních symbolů (tzv. vstupní abeceda).
- $\delta : \mathrm{Q} \times (\Sigma \cup \bar{\Sigma}) \rightarrow Q$ je přechodová funkce.
- $\mathrm{Q}_F \subseteq \mathrm{Q}$ je množina koncových stavů.
- $\mathrm{q}_0 \in \mathrm{Q}$ je počáteční stav.

Tento automat nazýváme deterministickým, pokud jsou splněny následující dvě podmínky.

- $|\delta(q, x)| \leq 1$ pro všechna $q \in Q$ a $x \in \Sigma \cup \bar{\Sigma}$.
- Pro každé $q \in Q$ a $a \in \Sigma$, jestliže $|\delta(q, a)| \neq 0$ nebo $|\delta(q, \bar{a})| \neq 0$ potom $|\delta(q, y)| = 0$ pro všechna $y \in (\Sigma \setminus \{\mathrm{a}\} \cup \bar{\Sigma} \setminus \{\bar{a}\})$. [5]

II. PRAKTICKÁ ČÁST

4 VÝVOJ APLIKACE

V praktické části práce se budeme zabývat vývojem desktopové aplikace pro simulaci všech variant multimnožinových konečných automatů. Hlavním cílem aplikace je jednoduché zadávání různých variant multimnožinových konečných automatů a přehlednou vizualizaci jejich činnosti. Aplikaci jsem pojmenoval **Simulátor multimnožinových konečných automatů** a je ve formátu spustitelného souboru.

4.1 Platforma, programovací jazyk a vývojové prostředí

Tato aplikace je primárně určena pro operační systém Windows 10 a program také dále běží na frameworku .NET 4.7.2. Pro práci s tímto frameworkem jsem zvolil programovací jazyk C#, k jehož psaní využívám vývojové prostředí Visual Studio 2019.

4.2 Systémové požadavky

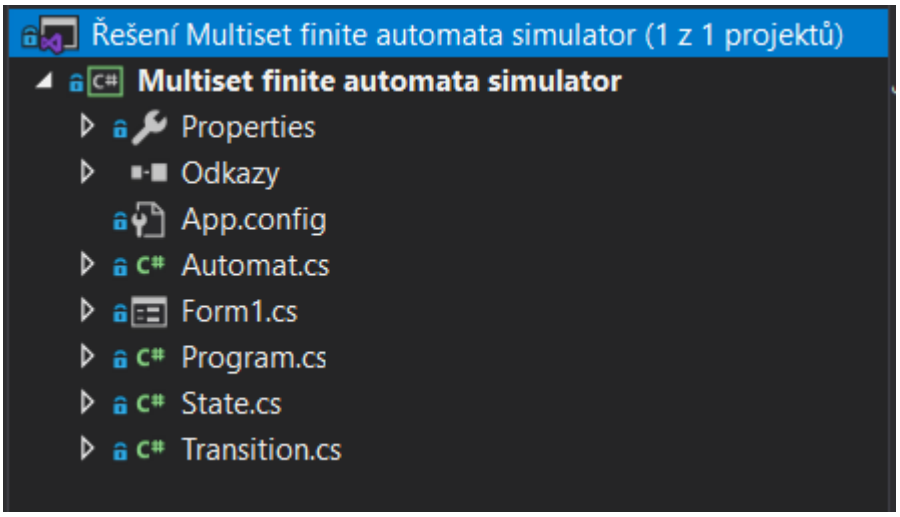
Minimálním požadavkem na operační systém je Windows 7. Na tomto operačním systému je však nutné doinstalovat framework .NET 4.7.2.

Doporučeným operačním systémem je Windows 10. Framework .NET 4.7.2 je již součástí tohoto systému.

4.3 Struktura programu

Program vytvořený v programovacím jazyku C# se může skládat z jednoho nebo více souborů. Tyhle soubory můžou obsahovat jmenné prostory, které obsahují prvky jako třídy, struktury, rozhraní, výčtové typy, ale také jiné jmenné prostory. [6]

Mnou vytvořená aplikace se také skládá z několika souborů. Tyto soubory jsou zobrazeny na *obrázku 8*.



Obrázek 8: Struktura programu

Na tomto obrázku se nachází celá struktura programu. Prvním důležitým souborem je *Multiset finite automata simulator*. Tento soubor se nazývá jmenný prostor. Ve jmenném prostoru se musí nacházet všechny třídy tohoto programu, jinak by tyto třídy nebyly navzájem viditelné.

Dalším důležitým souborem je soubor *Form1.cs*. Tento soubor obsahuje celé grafické rozhraní aplikace a je automaticky vytvořen při založení projektu. Dále se v souboru nachází veškeré metody všech ovládacích prvků a k těmto ovládacím prvkům lze přistupovat pouze z třídy, z které byly vytvořeny, tj. *Form1.cs*. Pro všechny ostatní třídy nejsou tyto ovládací prvky viditelné.

Následující třída je třída *Program.cs*, která je opět vygenerována při zakládání projektu. V této třídě se nachází pouze jedna jediná metoda, a to je metoda *Main*. Tahle metoda je tzv. vstupní bod a je to metoda, která se vykoná jako první po spuštění aplikace.

Nyní se dostáváme ke třídám, které nejsou vygenerovány automaticky při vytvoření projektu. První třída se nazývá *State.cs*. Tato třída slouží především pro ukládání důležitých informací o jednotlivých stavech multimnožinového konečného automatu. Mezi tyto informace patří název stavu a zda je tento stav koncový či počáteční. Dále je vhodné poznamenat, že třída *State.cs* implementuje rozhraní *ISerializable*.

Velmi podobnou třídou, třídě předchozí, je třída s názvem *Transition.cs*. Tato třída je navržena jako jeden přechod multimnožinového konečného automatu. Nejdůležitějšími

vlastnostmi této třídy jsou aktuální stav, čtený symbol a následující stav. Aktuální a následující stav jsou třídy *State.* Následně třída *Transition.cs* implementuje rozhraní *ISerializable*.

Poslední mnou vytvořenou třídou, je třída *Automat.cs*. Tato třída je bezesporu tou nejdůležitější, protože se zde odehrává výpočet. K výpočtu je potřeba znát tři hlavní parametry. Prvním parametrem je množina všech stavů, druhým parametrem je přechodová funkce a posledním třetím parametrem je vstupní multimnožina. Množina všech stavů je implementována jako *list* s parametrem *State*. Přechodová funkce je implementována velmi podobně, a to jako *list* s parametrem *Transition*. Vstupní multimnožina je implementována jako slovník, kde klíčem je znak symbolu a hodnota klíče je četnost. I tato třída implementuje rozhraní *ISerializable*. Všechny třídy, které implementují rozhraní *ISerializable*, jej implementují z důvodu ukládání těchto tříd na pevný disk.

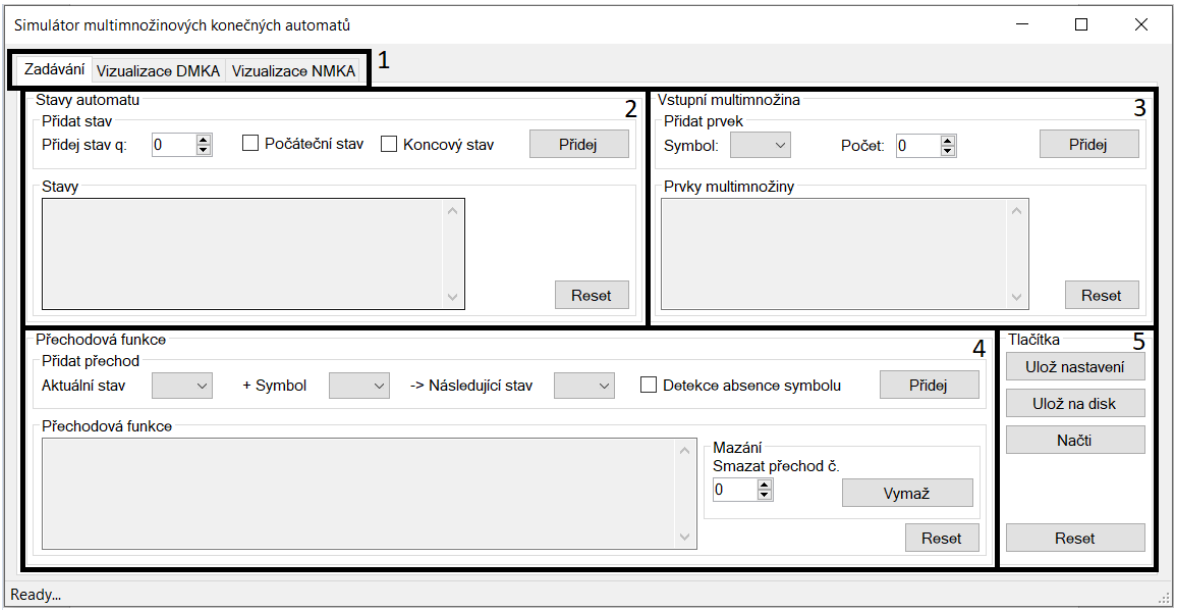
4.4 Uživatelské rozhraní

Tahle kapitola obsahuje podrobný popis všech jednotlivých částí uživatelského rozhraní a dále popisuje funkce jednotlivých tlačítek.

4.4.1 Hlavní okno

Na *obrázku 9* je naznačeno uživatelské rozhraní, které se zobrazí po spuštění aplikace. Po spuštění se aplikace nachází v záložce *Zadávání*.

Aby byl popis uživatelského rozhraní dostatečně srozumitelný, rozhodl jsem se rozčlenit uživatelské rozhraní do pěti sekcí.



Obrázek 9: Okno Zadávání

V sekci č. 1 se nachází tři záložky. Jednotlivé záložky slouží pro oddělení zadávání a samotnou vizualizací chodu multimnožinového konečného automatu.

Sekce č. 2, která se nazývá *Stavy automatu*, slouží k zadávání jednotlivých stavů multimnožinového konečného automatu.

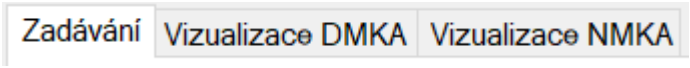
Následuje sekce č. 3, nazvaná *Vstupní multimnožina*, která je určena k zadání vstupní multimnožiny multimnožinového konečného automatu.

Přechodová funkce je pojmenovaná sekce s č. 4. Prostřednictvím čtvrté sekce je možné zadat přechodovou funkci multimnožinového konečného automatu.

Poslední sekce č. 5 se nazývá *Tlačítka*.

4.4.1.1 Záložky

Pro lepší přehlednost uživatelského rozhraní jsem se rozhodl oddělit zadávání automatu od samotné simulace výpočtu. K řešení tohoto problému jsem přistoupil tak, že jsem zadávání a simulaci výpočtu multimnožinového konečného automatu rozdělil do vlastních záložek. Tyto záložky jsou vyobrazeny na *obrázku 10*.



Obrázek 10: Záložky

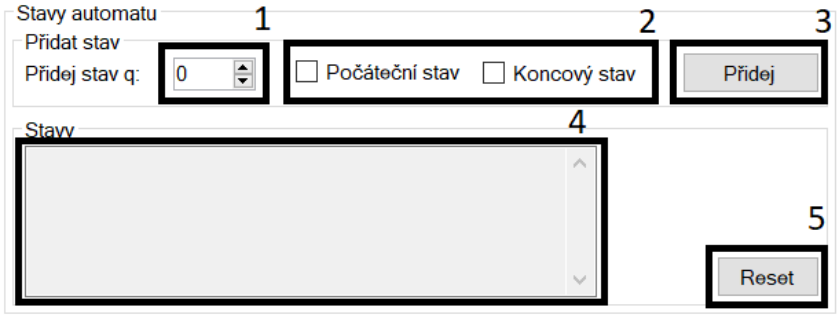
První záložka s názvem *Zadávání* slouží pro přepínaní mezi oknem pro zadávání jednotlivých parametrů multimnožinových konečných automatů.

Druhá záložka s názvem *Vizualizace DMKA* slouží k vizualizaci chodu deterministických konečných automatů.

Třetí záložka s názvem *Vizualizace NMKA* slouží k vizualizaci chodu nedeterministických konečných automatů.

4.4.1.2 Stavy automatu

Nedílnou součástí popisu multimnožinového konečného automatu jsou jeho stavy. Pro zadání jednotlivých stavů jsem vytvořil jednoduché okno, které je zobrazeno na *obrázku 11*. Pro lepší orientaci jsem vyznačil na obrázku 11 pět důležitých částí.



Obrázek 11: Zadávání stavů

V sekci č. 1 se nachází políčko pro zadávání čísel. Toto číslo určuje název stavu a je omezeno od nuly do sta. Číslo stavu můžeme vybrat kliknutím na šipky v pravé části políčka, nebo napsáním na klávesnici.

V sekci č. 2 jsou umístěny dvě zatrhávací pole. Prvním polem je *Počáteční stav* a druhým je *Koncový stav*. Tato pole určují, zda stav který chceme zadat, bude počáteční, koncový, nebo žádný z nich. Počáteční stav lze vybrat pouze jeden, ale koncových stavů může být několik.

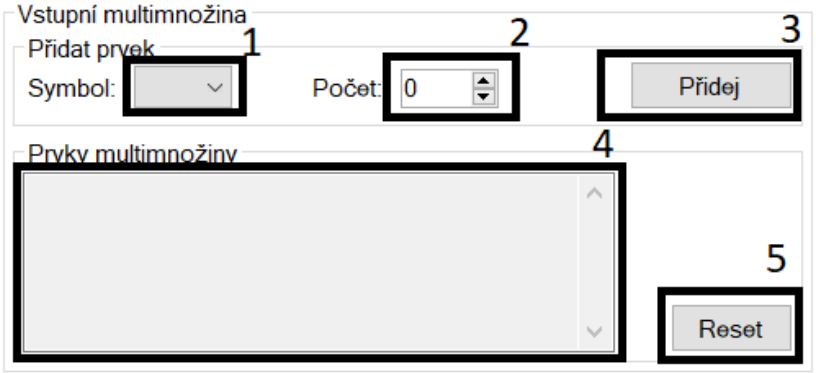
V sekci č. 3 se nachází tlačítko *Přidej*. Jak již název napovídá, tohle tlačítko přidává multimnožinovému konečnému automatu stavy. Kliknutím na tlačítko *Přidej* se číslo stavu spolu s informací o tom, zda je stav koncový, či počáteční, uloží do paměti. Dále po kliknutí na tlačítko *Přidej* se automaticky zvětší číslo stavu o jedno a zatrhávací políčka se vynulují.

V sekci č. 4 se nachází políčko, které slouží pouze jako výpis již zadaných stavů.

V sekci č. 5 vidíme tlačítko *Reset*. Tohle tlačítko slouží ke smazání všech již zadaných stavů.

4.4.1.3 Vstupní multimnožina

Dalším potřebným vstupním parametrem pro simulaci výpočtu multimnožinového konečného automatu je vstupní multimnožina. Zadávání symbolů vstupní multimnožiny je navrženo podobně jako zadávání stavů a je vyobrazeno na *obrázku 12*. Pro lepší orientaci jsem opět vyznačil pět důležitých částí.



Obrázek 12: Zadávání multimnožiny

V části č. 1 je umístěno pole pro výběr symbolu do multimnožiny. Do tohoto pole není možno psát, ale symbol lze vybrat kliknutím na pole. Po kliknutí na toto pole se zobrazí nabídka všech šestadvaceti symbolů anglické abecedy, z nichž je možno vybírat.

V části č. 2 se nachází pole pro zadání četnosti symbolu. Tato hodnota může nabývat hodnot od nuly do sta. Hodnota nula je zde zahrnuta z důvodu korektního výpočtu multimnožinových konečných automatů s detekcí.

V části č. 3 je zvýrazněno tlačítko *Přidej*. Tlačítko *Přidej* funguje obdobně jako u stavů automatu. Po kliknutí na tlačítko nastane uložení symbolu s jeho četností do paměti.

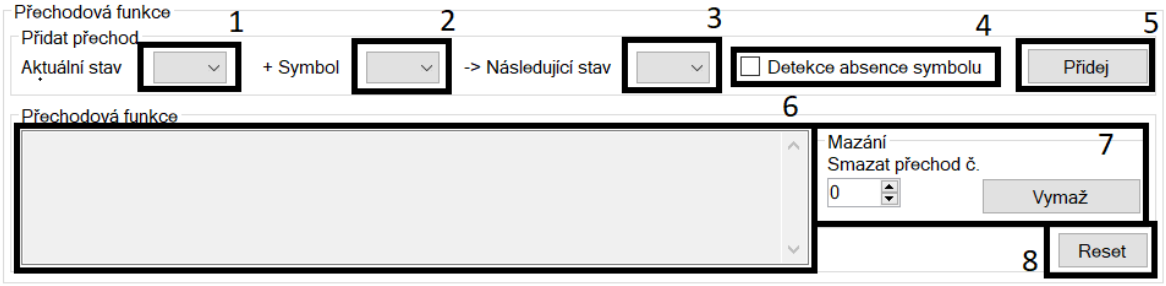
V části č. 4 se nachází pole, kde se vypisují informace o vstupní multimnožině.

V části č. 5 leží tlačítko *Reset*, které slouží pro vymazání vstupní multimnožiny.

4.4.1.4 Přechodová funkce

Poslední součástí popisu multimnožinového konečného automatu je přechodová funkce. Přechodová funkce se zadává pomocí jednotlivých přechodů. Jednotlivé přechody se skládají z aktuálního stavu, symbolu a následujícího stavu.

Pro zadávání všech hodnot jsem vytvořil uživatelské rozhraní, které je naznačeno na *obrázku 13*. Pro lepší popis tohoto uživatelského rozhraní jsem opět rozčlenil rozhraní do několika sektorů.



Obrázek 13: Zadávání přechodové funkce

V sektoru č. 1 se nachází políčko pro výběr aktuálního symbolu pro přidání přechodu. Do tohoto pole nelze psát, lze pouze vybrat z množiny již zadaných stavů multimnožinového konečného automatu.

V sektoru č. 2 vidíme podobné pole jako v předešlém sektoru. Jediným rozdílem je, že v tomto poli nevybíráme stavy, nýbrž symboly ze vstupní multimnožiny multimnožinového konečného automatu.

V sektoru č. 3 vidíme opět obdobné pole pro výběr. Tohle pole slouží k výběru následujícího stavu přechodu z množiny stavů multimnožinového konečného automatu.

V sektoru č. 4 leží zatrhávací okno *Detekce absence symbolu*, které slouží pro vytvoření přechodu pro multimnožinový automat s detekcí. Pokud tuhle možnost zatrhneme, tak se vytvořený přechod uskuteční jen tehdy, když četnost tohoto symbolu na vstupu bude nulová.

V sektoru č. 5 leží tlačítko *Přidej*. Z názvu je téměř jasné, že tlačítko přidá přechod do přechodové funkce.

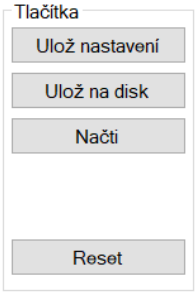
V sektoru č. 6 se nachází panel pro výpis přechodové funkce. Tento panel se aktualizuje vždy po kliknutí na tlačítko *Přidej* nebo *Vymaž*. Pro jednoznačnou identifikaci přechodů v přechodové funkci, je navíc každý přechod označen číslem.

V sektoru č. 7 se nachází možnost vymazání přechodu. V tomto sektoru se nachází dva objekty. V levé části tohoto sektoru se nachází vstupní pole, které slouží pro zadání čísla přechodu určeného ke smazaní. V pravé části sektoru se nachází tlačítko *Vymaž*. Kliknutím na toto tlačítko se vymaže přechod s číslem shodným s číslem z vedlejšího pole.

V sektoru č. 8 vidíme tlačítko *Reset*. Toto tlačítko vymaže celou přechodovou funkci.

4.4.1.5 Tlačítka

Pátou sekcí z *obrázku 9* je sekce s názvem *Tlačítka*. Tato sekce je pro připomenutí naznačena na *obrázku 14*.



Obrázek 14: Tlačítka

Tlačítko *Ulož nastavení* slouží k uložení zadaných hodnot do operační paměti. Aby tlačítko fungovalo korektně, musí být všechny vstupní parametry multimnožinového konečného automatu řádně vyplněny.

Následující tlačítko *Ulož na disk* má stejnou funkci jako tlačítko *Ulož nastavení*, jen s malým vylepšení. Toto vylepšení spočívá v tom, že nastavení automatu lze uložit kdekoliv na disk.

Tlačítko *Načti*, jak vyplývá z názvu, slouží k načtení souboru s parametry multimnožinového konečného automatu z disku do operační paměti. Po načtení tohoto souboru se parametry vyplní v celé záložce. Zadávání a parametry se můžou libovolně upravovat. Jakmile nastane úprava parametrů načtených ze souboru, je nutné tyto parametry

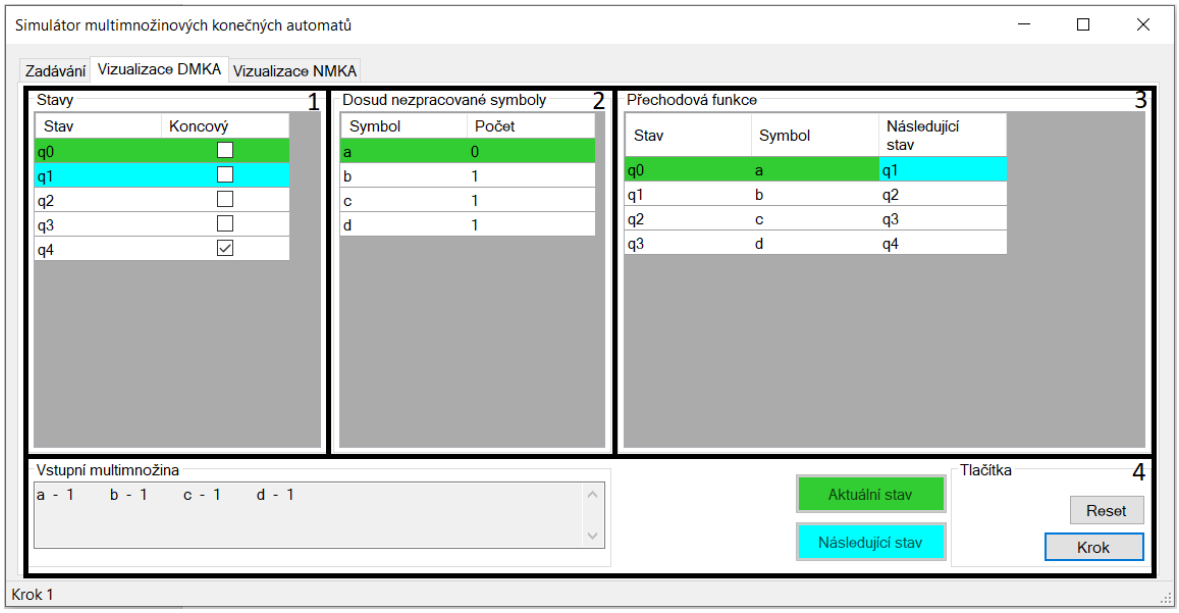
znovu uložit, ať už na disk, nebo do operační paměti, jinak program bude pracovat se starými parametry.

Posledním tlačítkem je tlačítko *Reset*. Kliknutím na toto tlačítko vymažeme všechna data v záložce *Nastavení*.

4.4.2 Okno simulace deterministického automatu

Nyní se od popisu záložky *Zadávání* dostáváme k popisu záložky *Vizualizace DMKA*. Tato záložka je určena pro vizualizaci výpočtu deterministického multimnožinového konečného automatu. Vizualizaci výpočtu jsem se rozhodl vytvořit pomocí tří tabulek. V těchto tabulkách jsou všechny důležité parametry deterministického multimnožinového konečného automatu. Celá vizualizace výpočtu funguje na principu vybarvovaní jednotlivých řádků tabulek dvěma různými barvami, a to zelenou a tyrkysovou.

Pro snadnější popis této záložky jsem se rozhodl, že ji opět rozdělím do několika částí, které jsou očíslovány v pravém horním rohu.



Obrázek 15 : Okno vizualizace DMKA

4.4.2.1 Stavy

V 1. části *obrázku 15* vidíme tabulku s názvem *Stavy*. Tato tabulka se skládá ze dvou sloupců. Pro každý jednotlivý stav je vyhrazen jeden řádek tabulky. V prvním sloupci se nachází název tohoto stavu. Stavy jsou zde pojmenovány tak, jak je zvykem, písmenem q a číslem stavu. Ve druhém sloupci se nachází informace, zda je tento stav koncový, či nikoliv. Koncový stav má vždy ve druhém sloupci zaškrtnuté okýnko. Pomocí této tabulky je naznačeno, ve kterém stavu se právě nachází, a který stav bude následující. Stav, ve kterém se deterministický multimnožinový konečný automat nachází je znázorněn zelenou barvou a následující stav je znázorněn barvou tyrkysovou.

4.4.2.2 Dosud nezpracované symboly

Ve 2. části *obrázku 15* se nachází tabulka s názvem *Dosud nezpracované symboly*. Tato tabulka se skládá opět ze dvou sloupců. Cílem tabulky je znázornit jakým způsobem deterministický multimnožinový konečný automat pracuje se vstupními symboly. Při každém výpočetním kroku se zelenou barvou označí aktuálně čtený symbol a sníží se hodnota jeho četnosti o jedna.

4.4.2.3 Přechodová funkce

Ve 3. části *obrázku 15* je tabulka s názvem *Přechodová funkce*. Tato tabulka je tvořena třemi sloupci. V prvním sloupci se nachází aktuální stav, ve druhém sloupci se nachází čtený symbol a v posledním sloupci se nachází následující stav. Jednotlivé řádky této tabulky mají význam jednotlivých přechodů. Tahle tabulka vyobrazuje aktuálně používaný přechod při konkrétním výpočetním kroku deterministického multimnožinového konečného automatu. Aktuálně používaný přechod se vybarvuje oběma barvami. Aktuální stav a aktuálně čtený symbol se vybarví barvou zelenou, přičemž následující stav vybarví barvou tyrkysovou.

4.4.2.4 Informační panel

Ve 4. části *obrázku 15* se nachází vícero objektů. V levé části je pole s názvem *Vstupní multimnožina*. Tohle pole je zde primárně pro to, aby po dokončení výpočtu bylo zřetelné,

jakou multimnožinu tento deterministický multimnožinový konečný automat přijímá, či nepřijímá.

Napravo od tohoto pole se nachází vysvětlení, co znamená zelená barva, a co znamená barva tyrkysová.

Poslední objekty, které se zde nachází jsou *Tlačítka*. Tlačítko *Krok* slouží k vizualizaci výpočetního kroku. Při stisknutí tohoto tlačítka se v tabulce *Stavy* vybarví aktuální a následující stav. V tabulce *Dosud nezpracované symboly* se vybarví aktuálně čtený symbol a jeho četnost se o 1 zmenší. V poslední tabulce s názvem *Přechodová funkce* se vybarví aktuálně používaný přechod. Tato situace se opakuje až do doby, kdy pro dvojici aktuální stav a symbol s nenulovou četností, existuje přechod.

Posledním tlačítkem je tlačítko *Reset*. Stisknutím tohoto tlačítka se resetuje celý výpočet a simulaci výpočtu je možno provést znovu.

4.4.3 Okno simulace nedeterministického automatu

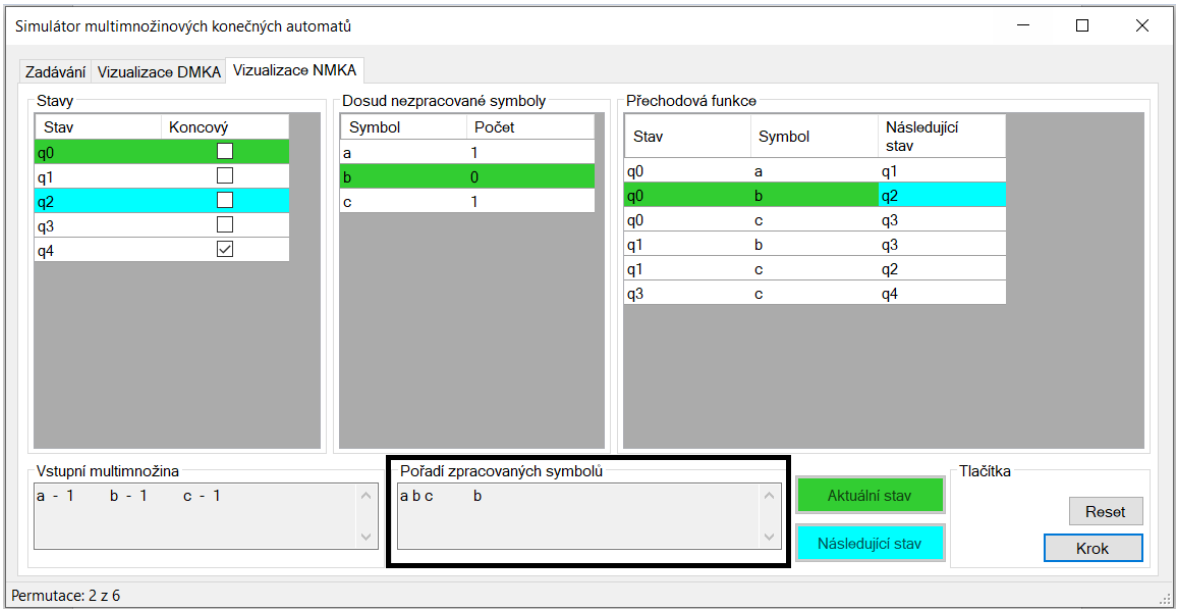
Nyní se přesouváme k poslední záložce *Vizualizace NMKA*. Tato záložka slouží k vizualizaci výpočtu nedeterministického multimnožinového konečného automatu. Hlavní myšlenkou při vývoji této záložky bylo to, aby se co nejméně lišila od záložky pro vizualizaci výpočtu deterministického konečného automatu, avšak nepatrný rozdíl zde přeci jenom nalezneme. Princip vizualizace výpočtu nedeterministického multimnožinového konečného automatu se mírně liší od jeho deterministické varianty.

Princip simulace výpočtu nedeterministického konečného automatu je takový, že ze symbolů vstupní multimnožinu vytvoří všechny možné permutace těchto symbolů. Tyto permutace vstupních symbolů jsou postupně přikládány na vstup nedeterministického multimnožinového konečného automatu.

S tímto řešením nastává problém, a to je počet symbolů vstupní multimnožiny. Počet všech možných permutací je tedy $n!$, kde n označuje počet různých vstupních symbolů. Např. pro $n = 5$ je počet všech možných permutaci 120.

Rozhodl jsem se proto tedy omezit symboly vstupní multimnožiny a to tak, že maximální počet různých symbolů bude 3 s tím, že jeden ze symbolů se může vyskytovat dvakrát. Při takto zvolené vstupní multimnožině je počet všech možných permutací 12.

Jelikož jsou záložky *Vizualizace DMKA* a *Vizualizace NMKA* téměř shodné, rozhodl jsem se na *obrázku 16* vyznačit pouze rozdílné pole *Pořadí zpracovaných symbolů*.



Obrázek 16: Vizualizace NMKA

Pole *Pořadí zpracovaných symbolů* slouží k postupnému výpisu jednotlivých permutací symbolů vstupní multimnožiny. Při každém úspěšném výpočetním kroku automatu se do tohoto pole vypíše aktuálně zpracovaný symbol. Pokud není žádný další přechod možný, celá permutace se zde vypíše a v dalším kroku nedeterministický multimnožinový konečný automat zpracovává další permutaci v pořadí.

5 NÁVOD NA OVLÁDÁNÍ

V této části práce se podrobně seznámíme jak s aplikací pracovat, jak správně multimnožinový konečný automat zadat a jak simulovat jeho výpočet.

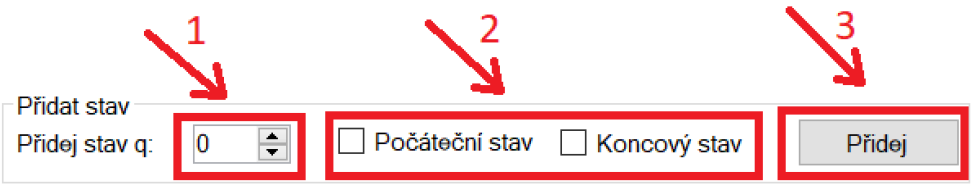
Jako první si spustíme aplikaci s názvem *Multiset finite automata simulator.exe*

5.1 Zadávání parametrů multimnožinového konečného automatu

Pro správné zadání multimnožinového konečného automatu jsou zapotřebí tyto parametry počáteční stav, koncový stav, vstupní multimnožinu a přechodovou funkci. V následujících krocích si podrobně ukážeme, jak všechny tyto parametry zadat. Všechny tyto parametry se zadávají v záložce *Zadávání*, která se zobrazí jako první po otevření aplikace.

5.1.1 Zadávání stavů

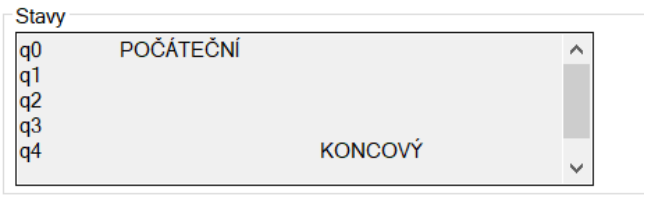
Jako první zadáme stavy automatu. Zadání stavů automatu probíhá ve třech krocích, které jsou na *obrázku 17*.



Obrázek 17: Pustup zadání stavu

Jako první krok si zvolíme název stavu. V následujícím kroku vybereme zda tento stav má být koncový či počáteční. V posledním kroku klikneme na tlačítko *Přidej*. Kliknutím na toto tlačítko se tento stav přidá. Tento postup opakujeme do té doby, dokud nemáme zadané všechny stavy multimnožinového konečného automatu.

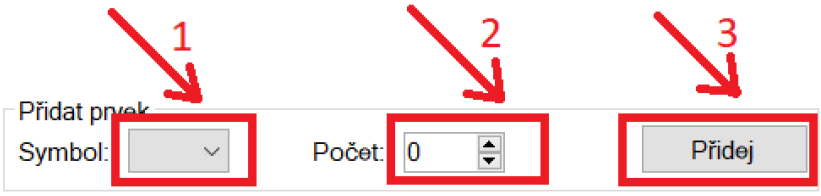
Zadání stavů multimnožinového konečného automatu může vypadat např. jako na *obrázku 18.*



Obrázek 18: Přehled stavů

5.1.2 Zadání vstupní multimnožiny

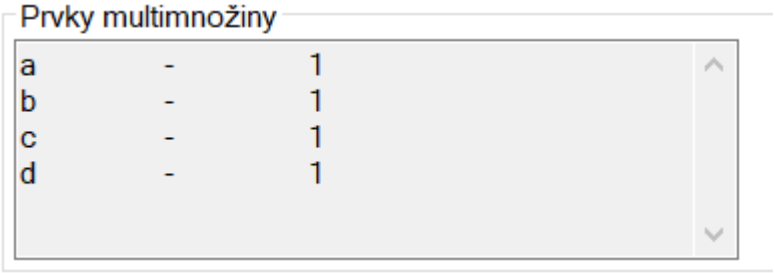
Jakmile máme zadány stavy multimnožinového konečného automatu, můžeme přejít na zadávání vstupní multimnožiny. Vstupní multimnožinu zadáme opět ve třech krocích, které jsou naznačeny na *obrázku 19*.



Obrázek 19: Postup zadání prvku multimnožiny

Prvním krokem je vybrání symbolu. Tento symbol vybereme tak, že klikneme na okno, které je označenou 1 na *obrázku 19*. Po kliknutí na toto okno se zobrazí nabídka všech možných symbolů a z této nabídky 1 symbol vybereme. Jakmile máme vybraný symbol, přesuneme se k dalšímu kroku, a to je výběr četnosti tohoto symbolu. Tato hodnota může být i 0. Jako poslední krok klikneme na tlačítko *Přidej*. Tento postup opět opakujeme do té doby, dokud nemáme zadané všechny symboly vstupní multimnožiny multimnožinového konečného automatu.

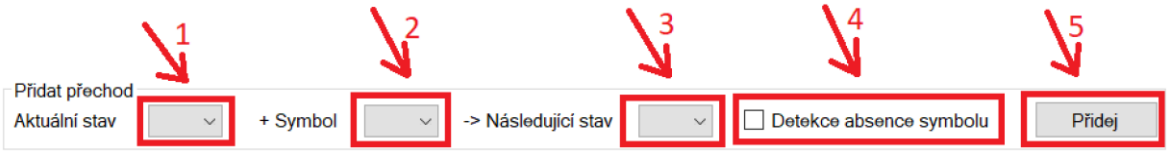
Výsledná multimnožinu může vypadat např. jako na *obrázku 20*.



Obrázek 20: Přehled symbolů multimnožiny

5.1.3 Zadání přechodové funkce

Posledním důležitým parametrem multimnožinového konečného automatu je přechodová funkce. Přechodovou funkci je nutno zadávat až jako poslední, protože se skládá ze stavů a symbolů. Přechodová funkce se zadává po jednotlivých přechodech. Toto zadávání je nastíněno na *obrázku 21*.



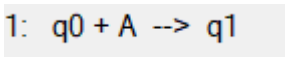
Obrázek 21: Postup zadání přechodu

První zadávací pole slouží k zadání aktuálního stavu. Po rozkliknutí se objeví nabídka, kde se vyskytují všechny stavy, které byly zadány v předchozím kroku.

Dalším zadávacím polem je pole pro zadávání symbolu.

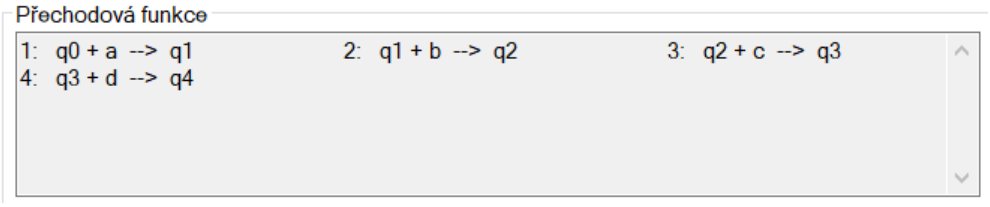
Třetí okno slouží pro výběr následujícího stavu.

Dalším důležitým prvkem, v pořadí čtvrtým, je zatrhávací okno s názvem *Detekce absence symbolu*. Tohle zatrhávací okno je zde proto, aby bylo možné simulovat výpočet multimnožinového konečného automatu s detekcí. Jakmile tohle okno zaškrtneme, automat vyhodnotí tenhle přechod pouze v případě, když je četnost tohoto symbolu 0. Pro odlišení od nedetekčního přechodu, je tento symbol znázorněn velkým písmenem. Podoba takového přechodu je zobrazena na *obrázku 22*.



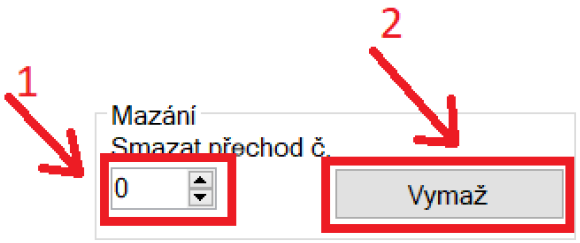
Obrázek 22: Přechod s detekcí absence symbolu

Jakmile máme vyplněna všechna pole, klikneme na tlačítko *Přidej*. Výsledná přechodová funkce může mít tvar jako na *obrázku 23*.



Obrázek 23: Přehled přechodové funkce

Na *obrázku 23* si můžeme všimnout, že každý přechod je očíslovaný. Toto číslování je z důvodu případného mazání přechodu. Postup pro mazání přechodu je nastíněn na *obrázku 24*.

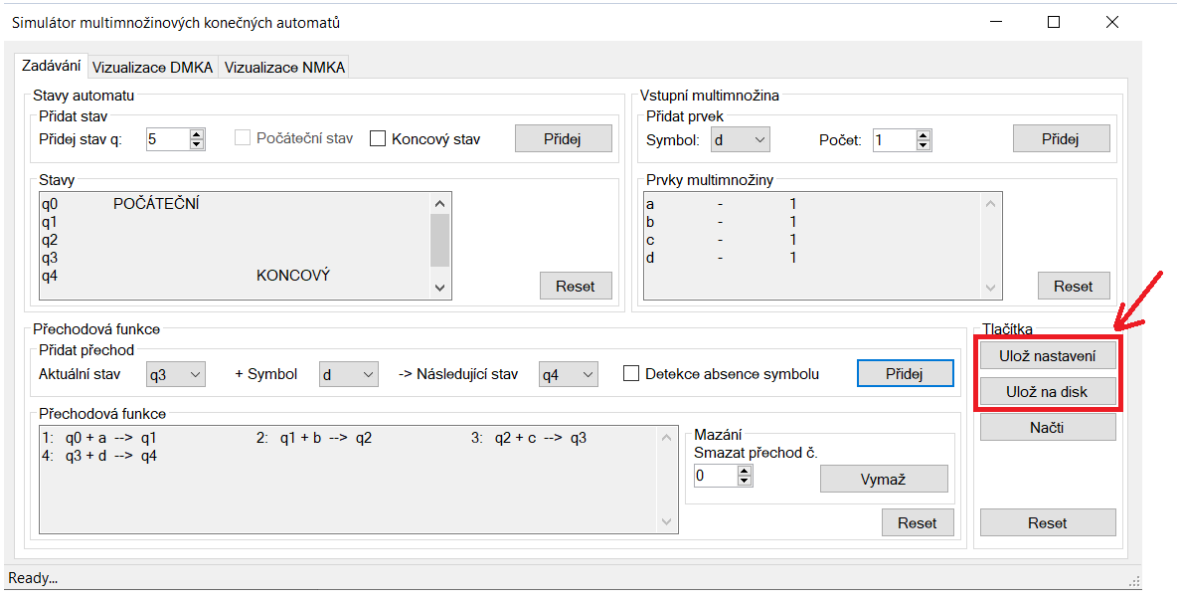


Obrázek 24: Postup mazání přechodu

Přechod, který má být vymazán, určuje číslo v prvním okně. Jakmile se toto číslo shoduje s číslem přechodu, můžeme kliknou na tlačítko *Vymaž*.

5.1.4 Ukládání automatu a načítání

Nyní již máme zadány všechny parametry, které jsou potřebné k vizualizaci výpočtu multimnožinového konečného automatu. Abychom mohli pokračovat k vizualizaci, je nutné tyto parametry uložit do paměti. Program disponuje uložením pouze do operační paměti, nebo uložením na pevný disk. Na *obrázku 25* jsou tyto tlačítka vyznačeny.

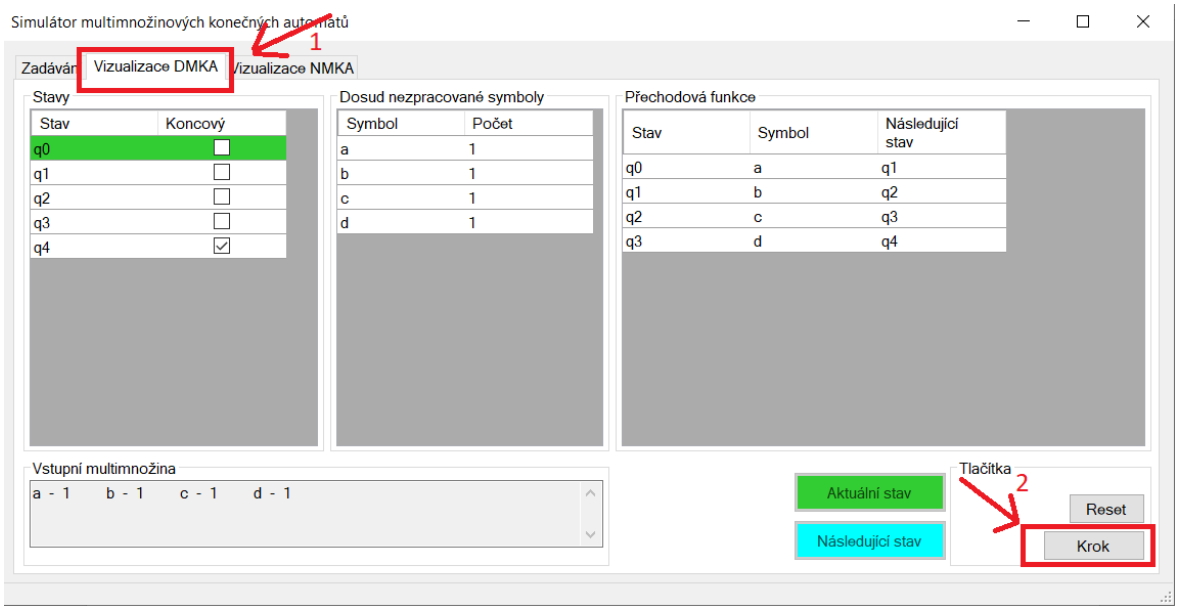


Obrázek 25: Uložení parametrů

Kliknutím na tlačítko *Ulož nastavení*. Se parametry uloží do operační paměti a po zavření programu se tato data zničí. Abychom s těmito parametry mohli pracovat i u příštího otevření programu, je nutné je uložit na disk. Pro tuto funkci slouží tlačítko *Ulož na disk*. Jakmile klikneme na tlačítko *Ulož na disk*, tak se otevře nové okno, které nás vyzve, kam si přejeme soubor s parametry uložit. Pokud parametry uložíme na disk, uloží se tyto parametry i do operační paměti a není potřeba klikat na tlačítko *Ulož nastavení*.

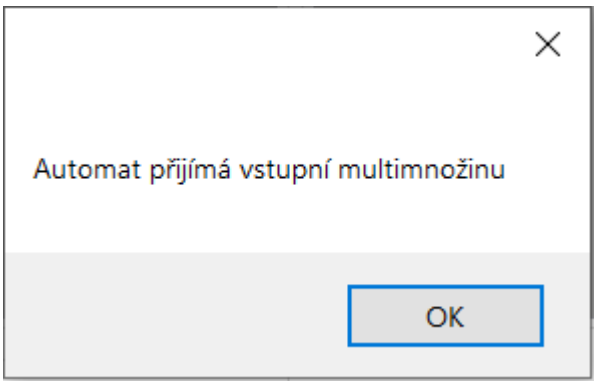
5.2 Vizualizace výpočtu

Jakmile máme multimnožinový konečný automat zadán, můžeme se vrhnout na vizualizaci výpočtu. Jako první věc se musíme přepnout do záložky *Vizualizace DMKA* nebo *Vizualizace NMKA*. Jakmile přepneme záložku, stačí už jen klikat na tlačítko *Krok*, do té doby dokud výpočet není u konce. Tento postup je naznačen na *obrázku 26*.



Obrázek 26: Postup vizualizace výpočtu

Jakmile projdeme celý výpočet, aplikace oznámí, zda multimnožinový konečný automat přijímá či nepřijímá vstupní multimnožinu. Toto oznámení je zobrazeno na *obrázku 27*.



Obrázek 27: Hláška o přijetí multimnožiny

ZÁVĚR

Výsledkem mé bakalářské práce je desktopová aplikace s názvem **Simulátor multimnožinových konečných automatů**, která je primárně určena pro operační systém Windows. Pro vývoj jsem použil vývojové prostředí *Visual Studio 2019*, kde se programuje v programovacím jazyce *C#*. Tento programovací jazyk jsem zvolil proto, protože si myslím že pro takovýto typ aplikace je *C#* nejvhodnější programovací jazyk. Aplikace umožňuje zadávání různých variant multimnožinových konečných automatů, ukládání těchto variant na pevný disk, načítání z pevného disku a simulaci výpočtu všech variant. Všechny tyto varianty jsou popsány v teoretické části, jak je určeno ze zadání.

Přínosem této práce je nejen výsledný software, ale i popis problematiky multimnožinových konečných automatů. Jelikož k tomuto tématu není zatím mnoho zdrojů, česky téměř žádný, tak si myslím, že někomu, kdo bude chtít toto téma studovat, tato práce studium usnadní.

Vývoj a testování aplikace byl velmi časově náročný, a to i navzdory tomu, že jsem s vývojem podobných aplikací už měl zkušenosti.

V budoucnu by se do aplikace dalo implementovat mnoho různých vylepšení, např. jako krok zpět ve vizualizaci výpočtu, nebo vizualizace výpočtu pomocí stavového diagramu. Tato vylepšení by umožnily širší začlenění aplikace do praxe.

SEZNAM POUŽITÉ LITERATURY

[1] KUDLEK, Manfred, Patrick TOTZKE a Georg ZETZSCHE. Multiset Pushdown Automata. *Fundamenta Informaticae* [online]. 2009, **2009**(93), 221-233 [cit. 2020-08-10]. DOI: 10.3233/FI-2009-98. Dostupné z: https://www.deepdyve.com/lp/ios-press/multiset-pushdown-automata-74PXORItAF

[2] MARTINEK, Pavel. *Some notes to minimization of multiset finite automata* [online]. Zlín: Department of Mathematics, Tomas Bata University in Zlin, 2018 [cit. 2020-08-10]. 10.1063/1.5044089. Dostupné z: https://aip.scitation.org/doi/abs/10.1063/1.5044089 Proceedings. 1978. 470019. 10.1063/1.5044089.

[3] *Základy teoretické informatiky* [online]. Pavel Martinek, 2006 [cit. 2020-08-10]. Dostupné z: http://phoenix.inf.upol.cz/esf/ucebni/zti.pdf

[4] *Automaty a formální jazyky I* [online]. Brno: Fakulta informatiky Masarykova univerzit, 2002 [cit. 2020-08-10]. Dostupné z: https://is.muni.cz/elportal/estud/fi/js06/ib005/Formalni_jazyky_a_automaty_I.pdf

[5] Csuhaj-Varjú E., Martín-Vide C., Mitrana V. (2001) Multiset Automata. In: Calude C.S., PĂun G., Rozenberg G., Salomaa A. (eds) Multiset Processing. WMC 2000. Lecture Notes in Computer Science, vol 2235. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45523-X_4

[6] *General Structure of a C# Program* [online]. Microsoft, 2015 [cit. 2020-08-10]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/general-structure-of-a-csharp-program

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

Tj. to je

Tzv. takzvaně, takzvaný

č. číslo

MKA multimnožinový konečný automat

DMKA deterministický multimnožinový konečný automat

NMKA nedeterministický multimnožinový konečný automat

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

PŘÍLOHA P I: ODKAZ NA ONLINE ZDROJOVÝ KÓD

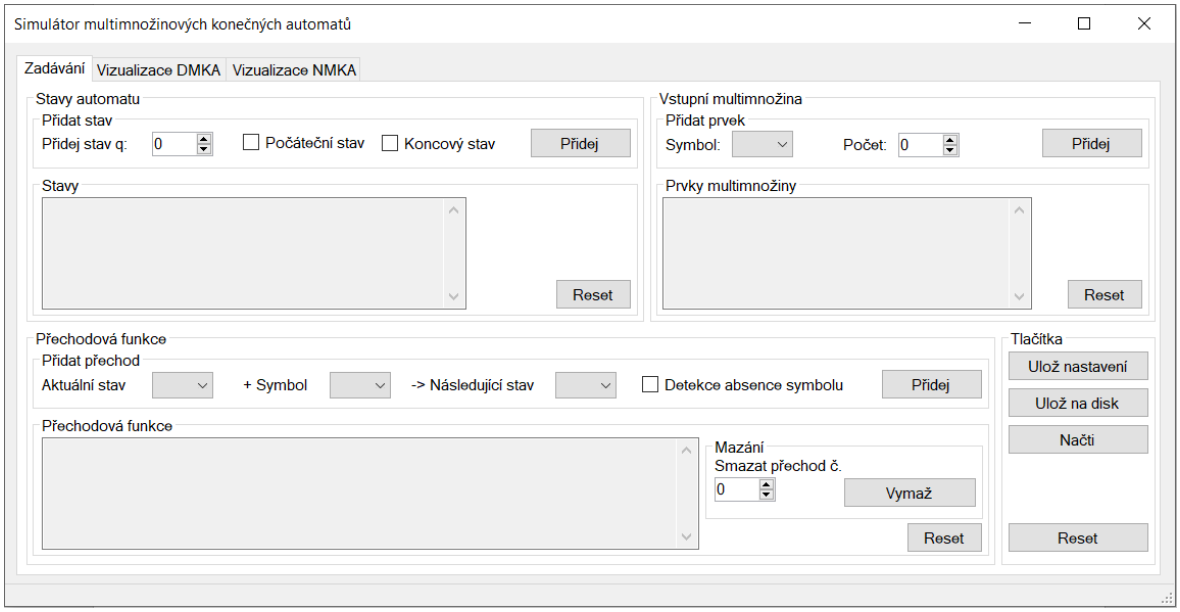
https://github.com/mpetrencak/MFASimulator

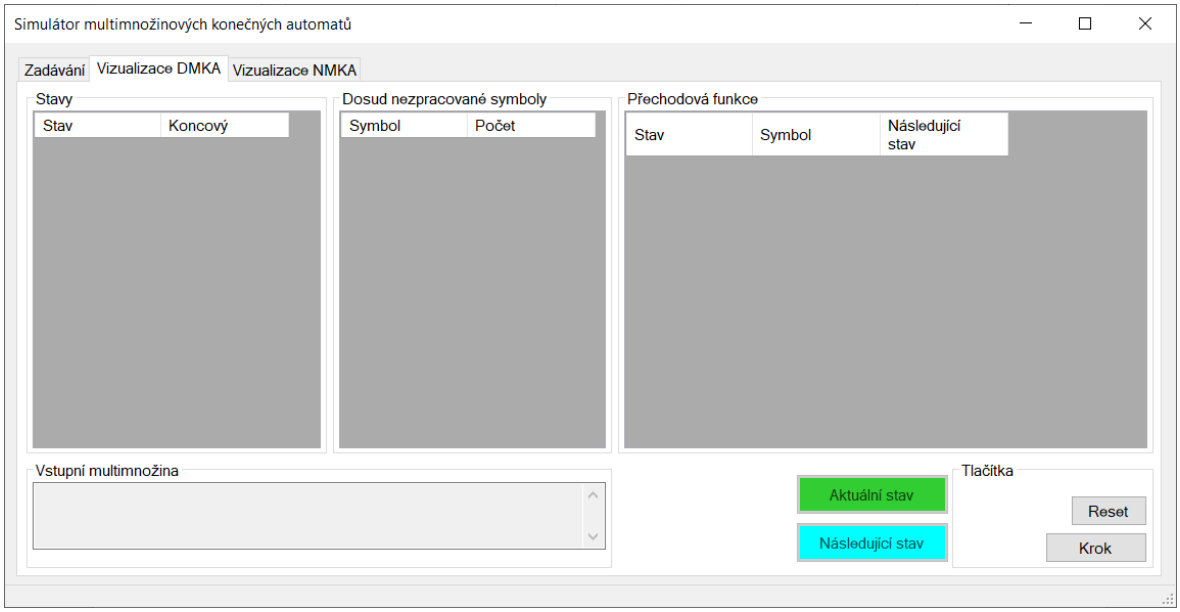
PŘÍLOHA P II: SOUBOR VE FORMÁTU ZIP

V soubor s názvem *prilohy.zip* je umístěn adresář s názvem *Multiset finite automata simulator*. Tento adresář obsahuje celý projekt.

Dále se zde nachází spustitelný soubor s názvem *Simulátor multimnožinových konečných automatů*.

PŘÍLOHA P III: GRAFICKÉ UŽIVATELSKÉ ROZHRANÍ





PŘÍLOHA P III: GRAFICKÉ UŽIVATELSKÉ ROZHRANÍ