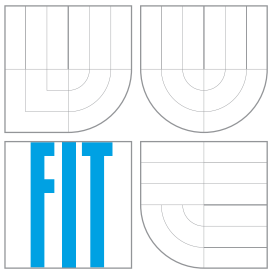
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

GLOBÁLNĚ ŘÍZENÉ CELULÁRNÍ AUTOMATY

GLOBALLY CONTROLLED CELLULAR AUTOMATA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MARTIN ŠVANTNER

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Ing. LUKÁŠ SEKANINA, Ph.D.

BRNO 2010

Abstrakt

Tato diplomová práce obsahuje úvod do problematiky celulárních automatů a podrobněji se zabývá možností jejich globálního řízení. Popisuje implementaci simulátoru globálně řízených celulárních automatů. Cílem je na základě provedených experimentů navrhnout a vyzkoušet klasifikaci celulárních automatů s globálním řízením. Návrh klasifikace je založen na ohodnocení vlivu jednotlivých parametrů globálně řízeného celulárního automatu na dynamiku chování automatu.

Abstract

This master's thesis deals with cellular automata and further deals possibility of their global control. It describes the implementation of simulator of globally controlled cellular automata. The goal is, design and test the classification of globally controlled cellular automata. Classification is based on evaluation of influence of each parameter of automata on their dynamics.

Klíčová slova

celulární automat, globální řízení, klasifikace, Wolframovy třídy

Keywords

Cellular Automaton, Global Control, classification, Wolfram classes

Citace

Martin Švantner: Globálně řízené celulární automaty, diplomová práce, Brno, FIT VUT v Brně, 2010

Globálně řízené celulární automaty

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Doc. Ing. Lukáše Sekaniny, Ph.D. a uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Švantner
23. května 2010

Poděkování

Tímto bych chtěl poděkovat panu Doc. Ing. Lukáši Sekaninovi, Ph.D. za odbornou konzultaci a připomínky při vypracování této diplomové práce.

Obsah

Kapitola 1

Úvod

Celulární automaty jsou jedním z biologií inspirovaných výpočetních modelů. Byly zformulovány Johnem von Neumannem a Stanislawem Ulamem v roce 1940, za účelem poskytnutí formálních konstrukcí pro vyšetřování chování komplexních a rozsáhlých systémů především biologického rázu, které jsou schopné sebereprodukce.

V článku Global Control In Polymorphic Cellular Automata [13] se objevila možnost globálního řízení celulárních automatů. V této variantě jsou pravidla celulárního automatu modifikována globálně a během simulace dochází k jejich změně.

Cílem této diplomové práce je zjistit jaké parametry ovlivňují chování globálně řízených celulárních automatů a pokusit se navrhnout způsob klasifikace globálně řízených celulárních automatů. Čtenář je v první kapitole obecně seznámen s problematikou celulárních automatů, jejich vlastnostmi, parametry a užitím. Druhá kapitola omezuje prostor obecně uvedených celulární automatů na jeden rozměr, formálně takovýto automat definuje a zavádí kvantitativní hodnocení dynamiky a klasifikaci celulárních automatů. Čtvrtá kapitola přidává k jednorozměrným automatům vlastnost globálního řízení a takovýto automat je opět formálně definován. Další kapitola krátce popisuje dostupné nástroje pro simulování celulárních automatů. Do šesté kapitoly je shrnut popis implementace vlastního simulačního nástroje. Požadavky na nový simulátor jsou nejprve specifikovány, poté následuje krátký úvod do implementačního jazyka a možnostech grafického rozhraní a následuje popis samotné implementace, nejdříve z pohledu uživatele a poté z pohledu programátora. Sedmá kapitola shrnuje možnosti klasifikace 256 pravidel základního jednorozměrného celulárního automatu a uvádí pravidla použitá v následujících kapitolách. Osmá kapitola je věnována cíli této diplomové práce, charakterizaci globálně řízených celulárních automatů. Charakterizace je založena na vizuální klasifikaci globálního chování automatu, které je věnována první část této kapitoly. Dále je uvedena série experimentů, při kterých se zjišťuje vliv parametrů automatu na dynamiku jeho chování, resp. klasifikaci do Wolframových tříd. V poslední části kapitoly je souhrn a vyhodnocení provedených experimentů. Diplomovou práci uzavírá diskuse nad otevřenými otázkami z uvedené problematiky a závěr, který shrnuje dosažené výsledky.

Tato diplomová práce navazuje na stejnojmenný Semestrální projekt, ze kterého jsou převzaty a částečně upraveny kapitoly o teoretickém úvodu do problematiky celulárních a globálně řízených celulárních automatů.

Kapitola 2

Celulární automaty

Inspirace k vytvoření celulárního automatu pochází z buněčných tkání, které mají podobnou strukturu jako z nich odvozený výpočetní model. V abstrakci celulárního automatu se buněčná tkáň stává diskrétním prostorem. Komplexní vnitřní stav buněk je redukován na číselné nebo symbolické stavové proměnné. Složitá pravidla interakce, která upravují časovou dynamiku buněk, jsou převedena na matematické funkce nebo pravidla, která určují, jak se bude stavová proměnná buňky měnit v čase, s přihlédnutím na stav sousedních buněk, začínající v počáteční konfiguraci celulárního prostoru.

2.1 Neformální definice

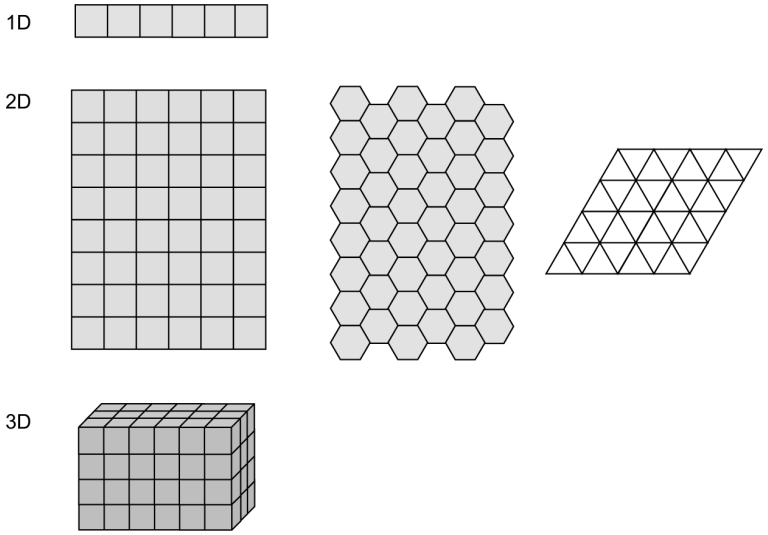
Neformálně je abstraktní celulární systém složen z následujících komponent [3]:

- celulární prostor,
- čas,
- stav a stavová proměnná,
- okolí,
- přechodová funkce,
- okrajové podmínky,
- počáteční stav a
- koncové podmínky.

Následující kapitoly popisují jednotlivé komponenty abstraktního celulárního systému.

2.1.1 Celulární prostor

Množina buněk v systému se nazývá celulární prostor. Obecně se jedná o pravidelnou n-rozměrnou mřížku buněk. V abstrakci celulárních automatů je tento prostor obvykle považován za nekonečný. V praxi je však nutné realizovat konečné rozměry prostoru. Obrázek 2.1 ukazuje některé běžné druhy celulárních prostorů. Celulární prostory s více než třemi rozměry se vyskytují jen velmi řídce, protože je nutno si uvědomit, že celkový počet buněk pro danou velikost roste exponenciálně s počtem dimenzí.



Obrázek 2.1: Běžně používané druhy prostoru [3].

2.1.2 Čas

Změny v celulárním systému se odvíjí podle časové osy, která může být diskrétní (většina případů) nebo spojitá.

2.1.3 Stav a stavová proměnná

Stav buňky systému reprezentuje stavová proměnná. Stavová proměnná může nabývat jednu hodnotu z konečné množiny přípustných hodnot buňky. Často se definuje speciální stav s_0, který představuje neaktivní stav buňky. Namísto jedné proměnné je možno použít n-tici proměnných. Každá z proměnných může představovat jiný parametr buňky.

2.1.4 Okolí

Okolí buňky je soubor buněk včetně samotné buňky, jejichž stav může přímo ovlivnit budoucí stav buňky.

Tvar a velikost okolí nejsou nijak striktně definovány, nicméně okolí se skládá z malého počtu okolních buněk, protože celulární systémy jsou považovány za modely systémů, ve kterých se informace vyměňují pouze lokálně.

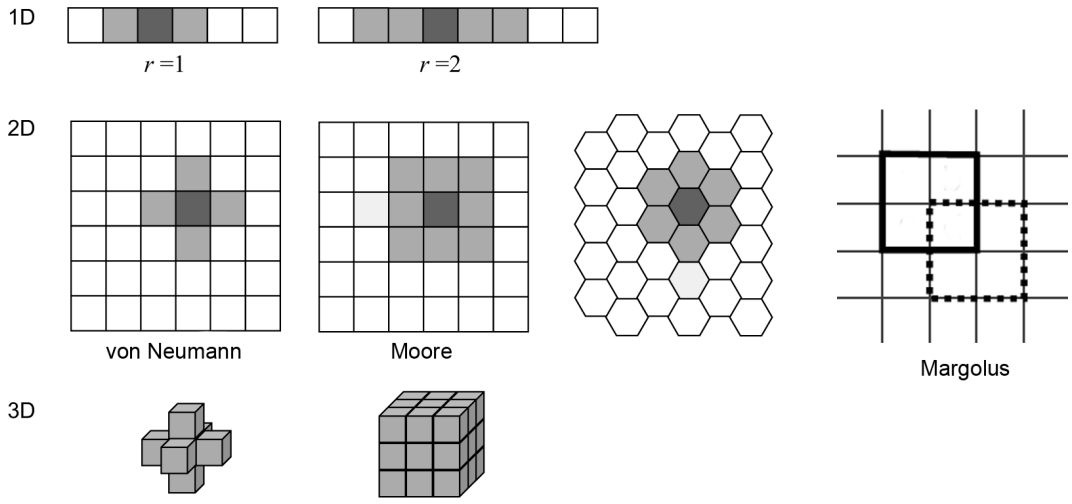
Jednoduchým způsobem, jak definovat okolí, je definovat vzdálenost r v celulárním prostoru a následně definovat, že okolím buňky jsou všechny buňky s menší vzdáleností než je zadána.

Obrázek 2.2 zobrazuje několik nejčastěji používaných druhů okolí v závislosti na počtu dimenzí prostoru [9].

Pokud mají všechny buňky celulárního systému stejný druh okolí, hovoříme o homogenním nebo uniformním prostředí.

2.1.5 Přechodová funkce

Přechodová funkce definuje, jak se stav buňky mění v čase. Záleží na současném stavu buňky, stavu buněk v okolí a případně na pozici buňky v celulárním prostoru.



Obrázek 2.2: Běžně používané druhy okolí - pro jednorozměrný prostor je použita definice vzdálenosti r, ve dvou-rozměrném prostoru se jedná především o von Neumannovo okolí, které se nazývá také čtyř-okolí a Mooreovo okolí, neboli osmi-okolí [3][9].

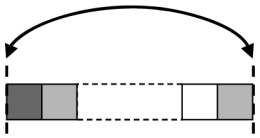
Pokud mají všechny buňky prostoru stejnou přechodovou funkci nebo přechodová funkce není závislá na čase, nazývá se celulární systém homogenní vzhledem k přechodové funkci, v čase resp. prostoru. Obecně, pokud je celulární systém nazván homogenní, bez další specifikace, rozumí se homogenní vzhledem k okolí a přechodové funkci, resp. v prostoru a čase.

Pro v čase diskrétní systémy je možno přechodovou funkci na počítači implementovat jako programovou rutinu (metodu, proceduru,..), která vyhodnotí nový stav buňky v každém časovém kroku. Pro malé konečné stavové množiny a malé okolí se dá přechodová funkce implementovat vyhledávací tabulkou, ve které jsou uloženy všechny vstupy a k nim příslušné výstupy.

2.1.6 Okrajové podmínky

Pokud je celulární prostor konečný, vyskytují se v něm buňky, pro které je nutno specifikovat okrajové podmínky. Jsou to takové buňky, u kterých nelze definovat celé okolí. Existuje několik běžně používaných okrajových podmínek:

- **Periodické** podmínky jsou jedny z nejjednodušších. Řešením okrajových podmínek je transformovat celulární systém s hranicemi na celulární systém bez hranic. Standardní, obdélníkový dvourozměrný celulární prostor se transformuje na toroidní celulární prostor, který vznikne spojením opačných stran obdélníkového prostoru. Toto řešení, které je uvedeno na obrázku 2.3, se nazývá *periodické okrajové podmínky.*

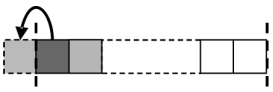


Obrázek 2.3: Periodické okrajové podmínky [3].

- **Pevné** podmínky jsou další z možností. Řešením okrajových podmínek je definice virtuálního okolí. Virtuální buňky, které jsou potřebné k získání okolí, mohou mít přiřazen stav, nezávisle na aktuálním stavu celulárního systému. Ve většině případů je přiřazený stav fixní (*pevné okrajové podmínky*), ale může být také generován složitější funkcí, například náhodnými čísly (*náhodné okrajové podmínky*) nebo může být navázán na některou ze sledovaných veličin celulárního systému. Pevné okrajové podmínky jsou zobrazeny na obrázku 2.4.

Obrázek 2.4: Pevné okrajové podmínky [3].

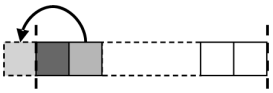
- **Adiabatické** podmínky jsou alternativou k pevným okrajovým podmínkám. Řešením je kopírování stavu jiné buňky celulárního systému. Adiabatické okrajové podmínky jsou definovány pomocí kopírování stavu hraničních buněk systému.



Obrázek 2.5: Adiabatické okrajové podmínky [3].

Název je odvozen od systémů používaných k modelování rozptylu tepelných jevů, které mají teplotu reprezentovánu jako stav buňky. U těchto systémů tato strategie definuje nulový teplotní gradient a tím nulovou výměnu tepla přes hranice systému. Tato strategie je zobrazena na obrázku 2.5.

Zrcadlové okrajové podmínky využívají k definování virtuálního okolí kopii buňky, která je umístěna jako další za okrajovými buňkami.

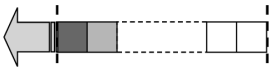


Obrázek 2.6: Zrcadlové okrajové podmínky [3].

- **Otevřené hranice** nebo také otevřené okrajové podmínky jsou speciální třídou podmínek, která umožňuje simulovat v konečném prostoru chování celulárních systémů s nekonečným prostorem.

Cílem je na hranici celulárního prostoru definovat proces, který zajistí neporušení aktivit v konečném prostoru, ve kterém je modelován. Definice tohoto procesu závisí na podrobnostech definice přechodové funkce a může to být obtížné.

Idea této strategie je zobrazena na obrázku 2.7.



Obrázek 2.7: Otevřené hranice

2.1.7 Počáteční stav

Pro úspěšné spuštění celulárního systému v souladu s přechodovou funkcí je nutné definovat počáteční stav všech buněk systému. Toto přiřazení může být nazváno také *počáteční podmínky*.

2.1.8 Koncové podmínky

Ukončující podmínky specifikují, za jakých podmínek má být ukončen cyklus aktualizace celulárního prostoru. Nejčastější koncová podmínka je předem definovaná délka simulačního času.

2.2 Aplikace celulárních automatů

Následující kapitola popisuje nejčastěji uváděný příklad celulárních automatů, hru Life a dále shrnuje některé aplikace celulárních automatů.

2.2.1 Game of Life

Game of Life (Conway's Game of Life), v překladu *Hra života* [8] vytvořil matematik John Horton Conway, který byl posedlý myšlenkou sestavení jednoduchého dvourozměrného celulárního automatu. Po dlouhých experimentech došel k řešení, které uvažuje dva stavy buňky:

- **živá buňka** a
- **neživá buňka**.

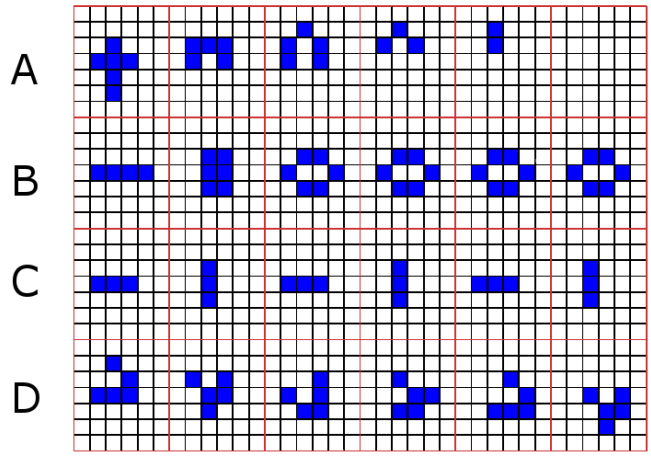
Takto definované stavy buněk budou uvažovány pro další popis. Pro výpočet nového stavu buňky je uvažováno von neumannovské (osmi) okolí a následující pravidla pro přechodovou funkci:

- **zrod** - jsou-li v okolí prázdného políčka právě tři živé buňky, zrodí se nová živá buňka, neboli „trojpohlavní" rozmnožování,
- **přežití** - pokud jsou v okolí živé buňky dvě nebo tři další živé buňky, buňka přežívá,
- **uhynutí** - v případě, že v okolí je žádná nebo jedna živá buňka, buňka umírá osamocením; pokud jsou v okolí čtyři až osm živých buněk, umírá nedostatkem prostoru.

Takto definovaný celulární automat byl nazván hrou života, která se po uveřejnění v roce 1970 stala velmi populární.

Na obrázku 2.8 jsou zobrazeny jednoduché struktury ze hry Life. Do těchto struktur, po několika generacích, směřuje většina populací buněk. Struktury jsou rozděleny do čtyř kategorií, podle způsobu chování:

- **Zánik** - kategorie A.
- **Stabilní obrazec** - kategorie B, v dalších krocích neměnný.
- **Cyklicky se opakující obrazec** - kategorie C.
- **Cyklicky se opakující posunutý obrazec** - kategorie D.



Obrázek 2.8: Jednoduché struktury [27].

Kluzák a kluzákové dělo
Kluzákové dělo je jeden z pojmenovaných celulárních automatů založených na hře života. Na obrázku 2.9 je v horní části kluzákové dělo, tedy shluk buněk, které v každé 30. generaci generuje jeden kluzák. Ve spodní části obrázku jsou jednotlivé kluzáky, což jsou shluky buněk, které se pohybují po úhlopříčce celulárního prostoru.

Tento model je vázán na pokus dokázat, že celulární automat tvoří universum, do kterého je možno vsadit Turingův stroj. Z tohoto hlediska se stalo klíčovou otázkou existence obrazce, který by se přemisťoval a tím se z něj stal nosič informace. A existence druhého obrazce, který by generoval uvedené pohyblivé obrazce. Tímto jsou kluzák a kluzákové dělo.

Obrázek 2.9: Kluzákové dělo [26]

2.2.2 Generátory pseudonáhodných čísel

Další z možností využití celulárních automatů jsou generátory pseudonáhodných čísel [7], ve kterých se využívá především pravidel ze třetí Wolframovy třídy. Výběr vhodného pravidla není jednoduchý úkol a neexistuje pro to spolehlivá metodika. Nejčastěji se pravidla vybírají pokusy nebo na základě předchozích zkušeností. Existuje několik málo metod, které

umožňují alespoň částečně vlastnosti generátoru odhadnout. Mezi vyzkoušená pravidla je možno zařadit pravidlo 30 a 50745. Vysvětlení významu čísel pravidel a tříd je uvedeno v následujících kapitolách.

2.2.3 Další aplikace celulárních automatů

Mezi další používané aplikace celulárních automatů můžeme zařadit [12]:

- generování testovacích vektorů,
- modely číslicových systémů (Cell-Matrix apod.),
- simulace systémů s lokálními interakcemi (např. požár lesa),
- simulace chování plynů,
- studium feromagnetismu,
- systémy reprezentované diferenciálními rovnicemi,
- simulace růstu krystalů,
- difuze tepla a znečištění,
- turbulentní proudění,
- modelování ekonomických procesů,
- modely v biologii a ekologii,
- v oblasti grafiky (textury),
- studium gramatik v teorii formálních jazyků.

Kapitola 3

Jednorozměrné CA

Tato kapitola je zaměřena na konkrétnější celulární automat a sice na jednorozměrný (1D CA), který bude také formálně definován.

3.1 Formální definice

Celulární automat [6] je nekonečná mřížka konečných stavových automatů, které se nazývají **buňky**. Buňky d-rozměrného celulárního automatu jsou umístěny na celočíselné mřížce bodů d-rozměrného Eukleidovského prostoru a jsou adresovány prvky z množiny $\mathbb{Z}^d$.

Definice 3.1.1 *Jednorozměrný binární* ***neuniformní*** *celulární automat s konečným počtem buněk je sedmice $A = (Q, N, R, z, b_1, b_2, c_0)$, kde: [11]*
$Q = \{0,1\}$ je binární množina stavů,
N určuje okolí ($N \subseteq \mathbb{Z}$)
z určuje počet buněk,
b_1 a b_2 jsou okrajové hodnoty,
c_0 je počáteční konfigurace, a
zobrazení $R : S \to (Q^N \to Q)$ přiřazuje každé buňce mřížky $S = \{1, 2, \ldots, z\}$ lokální přechodovou funkci $\delta_1 \ldots \delta_z$, kde $\delta_i : Q^N \to Q, i \in S$

Konfigurace automatu A je zobrazení $c \in Q^S$, které přiřazuje stav každé buňce automatu A. Pokud budeme uvažovat pouze jednoduché okolí, tedy $N = \{-1, 0, 1\}$ (definováno poloměrem $r = 1$), potom *globální přechodová funkce* $G : Q^S \to Q^S$ je definována:

$$G(c(i)) = \begin{cases} \delta_i(c(i-1), c(i), c(i+1)) & pro\ i = 2 \ldots z-1, \\ \delta_1(b_1, c(1), c(2)) & pro\ i = 1, \\ \delta_z(c(z-1), c(z), b_2) & pro\ i = z, \end{cases}$$

kde c_i označuje konfiguraci celulárního automatu v kroku i.

Globální přechodovou funkci G je možno použít k definování sekvence konfigurací $c_0, c_1, c_2, \ldots$ tak, že $c_j = G(c_{j-1})$, pro $j \geq 1$. Tato sekvence reprezentuje výpočet automatu A.

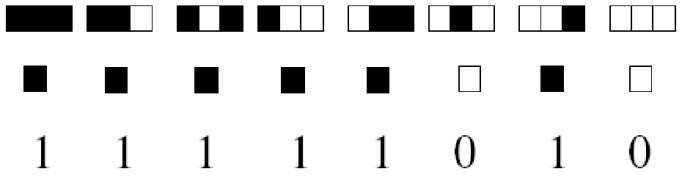
Dále uvažované celulární automaty jsou homogenní, neboli uniformní, které popisuje upravená předchozí definice:

Definice 3.1.2 *Jednorozměrný binární* ***uniformní*** *celulární automat s konečným počtem buněk je sedmice $A = (Q, N, \delta, z, b_1, b_2, c_0)$, kde*
δ je lokální přechodová funkce, $\delta : Q^N \to Q$
ostatní složky mají stejný význam jako u ne-uniformního automatu.

Konfigurace homogenního automatu je obdobná jako u ne-uniformního, pouze je nutno uvažovat, že lokální přechodová funkce je pro všechny buňky stejná.

3.2 Číslování pravidel

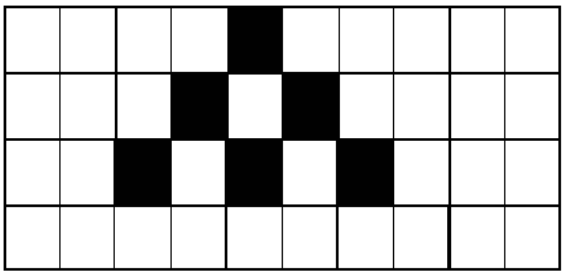
Pro popis pravidel jednorozměrného celulárního automatu je použito jedno dekadické číslo. Následující obrázek ukazuje pro větší přehlednost odvození pravidla pro 1D CA s okolím o velikosti tři buňky [12].



Obrázek 3.1: Ukázka pravidla 250 [12].

První řádek na obrázku 3.1 reprezentuje současný stav okolí. Prostřední buňka v trojici je aktuální buňka, pro kterou se počítá nový stav. Na druhém řádku je nový stav buňky. Třetí řádek reprezentuje nový stav číselně. Nechť jednotlivé číslice představují hodnoty řádů čísla ve dvojkové soustavě, potom celulární automat pracuje podle pravidla

$$11111010_b = 250_d.$$

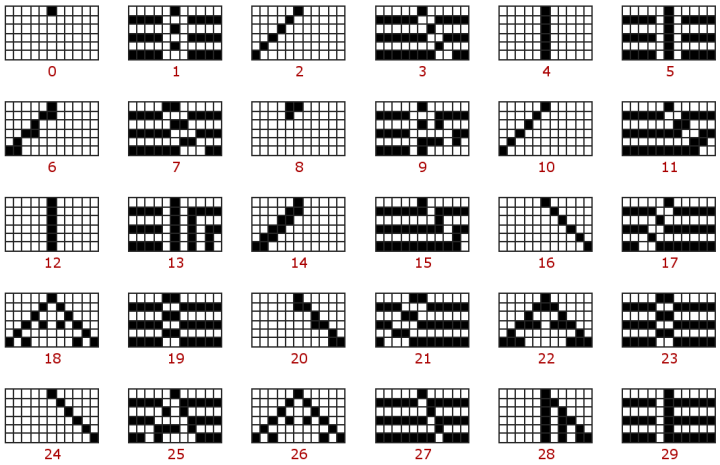


Obrázek 3.2: Aplikace pravidla 250

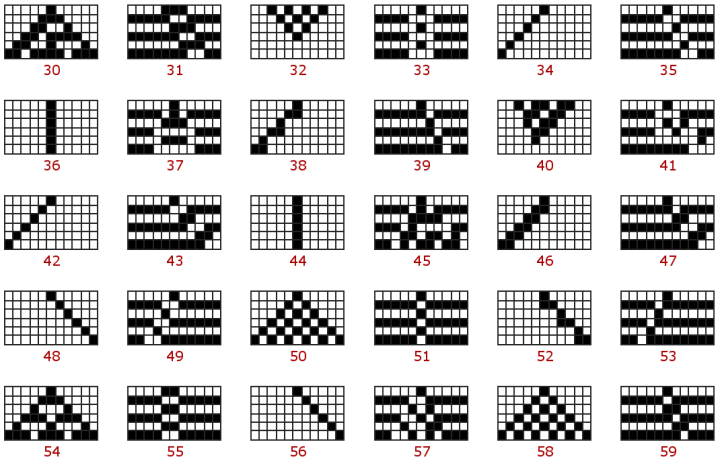
Obrázek 3.11 ukazuje vývoj 1D CA s výše definovaným pravidlem 250. První řádek je počáteční stav automatu a směrem dolů jsou stavy v následujících časových krocích.

3.3 Vývoj 1D CA

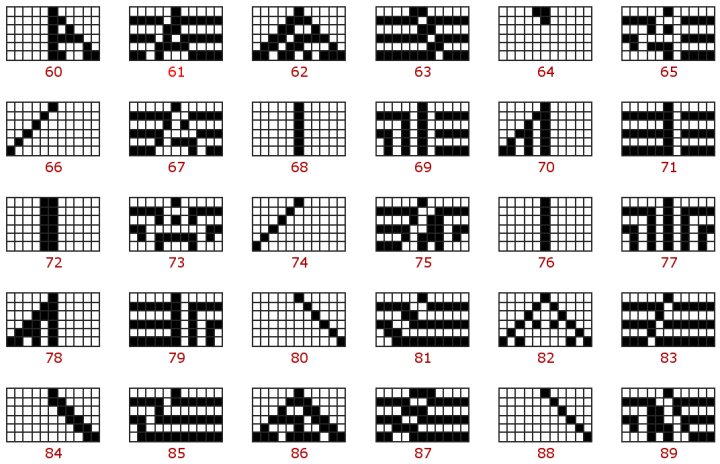
Do této kapitoly byly zařazeny průběhy všech 256 možných celulárních automatů proacujících podle výše popsaných pravidel a v následující kapitole budou tato pravidla rozdělena do Wolframových tříd, které charakterizují chování celulárního automatu. Počáteční konfigurace je vždy $\ldots 00100 \ldots$ a je uvedeno 6 kroků výpočtu automatů.



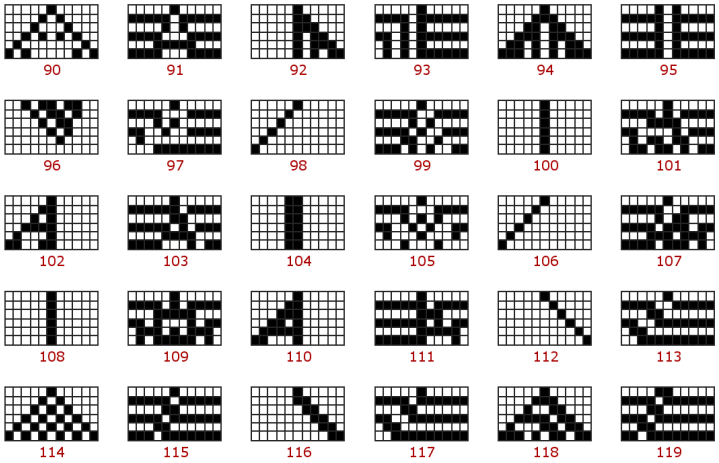
Obrázek 3.3: Pravidla 0 - 29 [22].



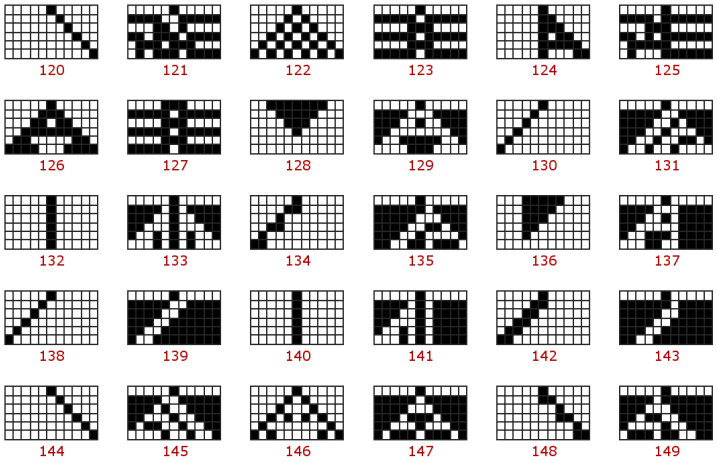
Obrázek 3.4: Pravidla 30 - 59 [22].



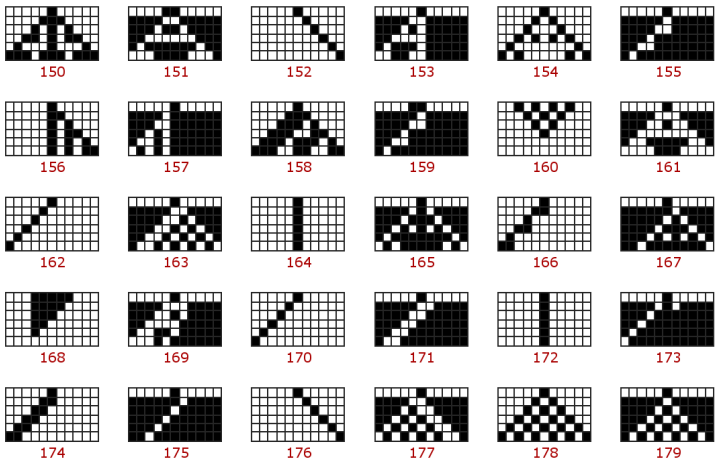
Obrázek 3.5: Pravidla 60 - 89 [22].



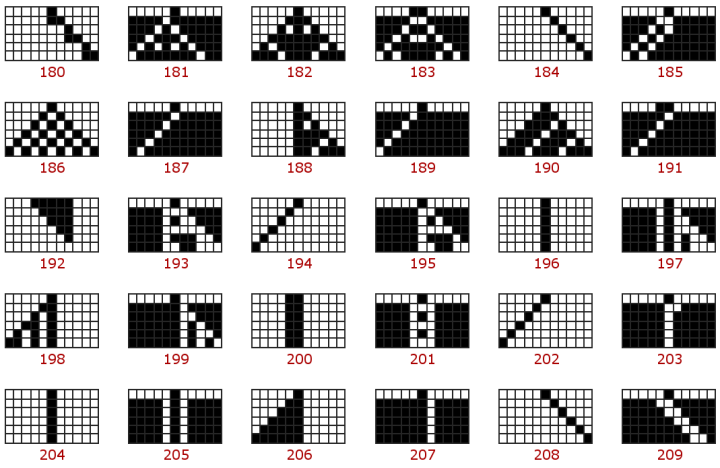
Obrázek 3.6: Pravidla 90 - 119 [22].



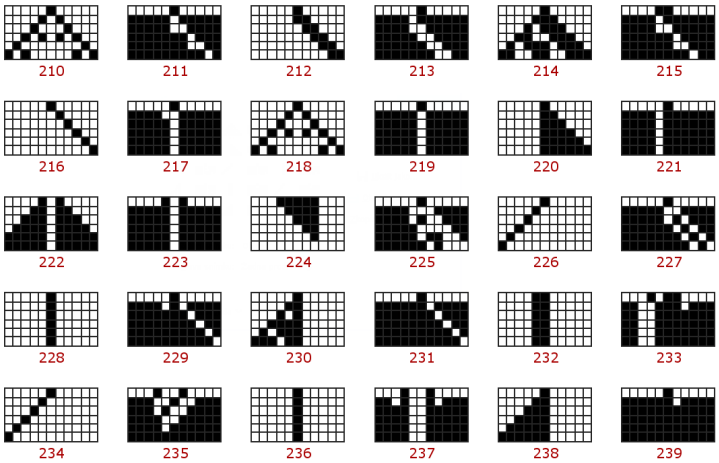
Obrázek 3.7: Pravidla 120 - 149 [22].



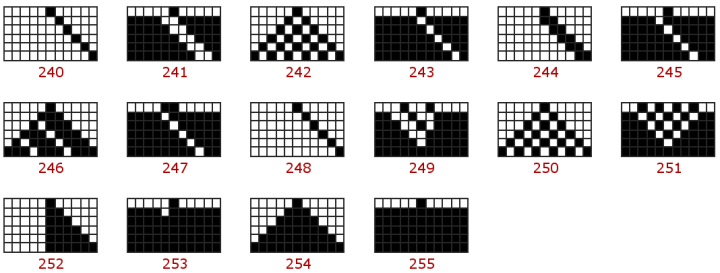
Obrázek 3.8: Pravidla 150 - 179 [22].



Obrázek 3.9: Pravidla 180 - 209 [22].



Obrázek 3.10: Pravidla 210 - 239 [22].



Obrázek 3.11: Pravidla 240 - 255 [22].

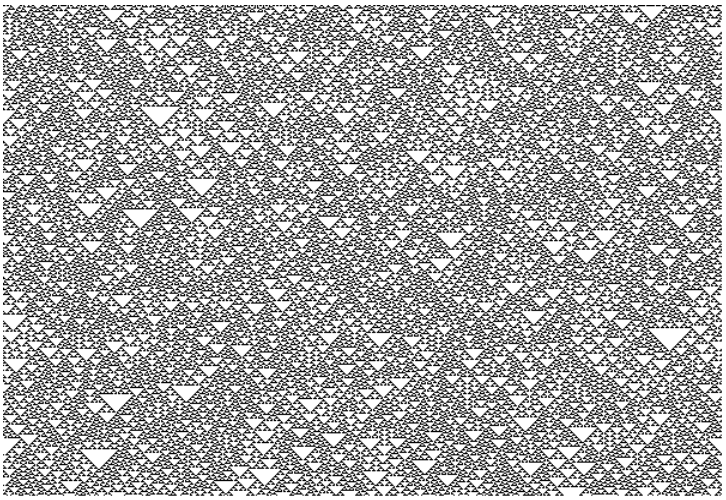
3.4 Wolframovy třídy

Stephen Wolfram [8] studoval jednorozměrné celulární automaty, protože jejich výhodou je poměrně malý počet možných pravidel a reprezentace průběhů v přehledné tabulce. Wolfram rozdělil uvedených 256 pravidel do čtyř skupin podle složitosti chování (definoval je sice pro 1D CA, ale ekvivalentní třídy se vyskytují i u vícerozměrných automatů).

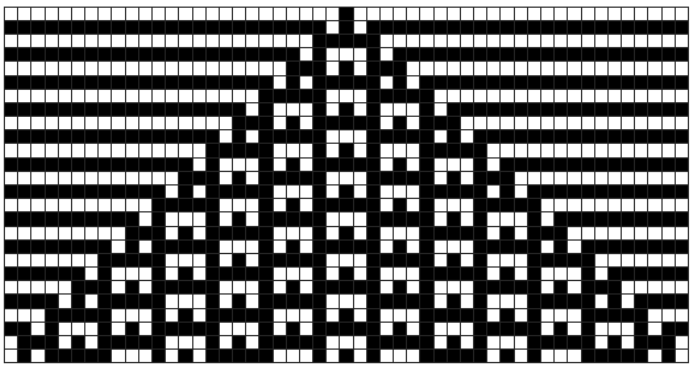
Wolframovy třídy:

- **CA1** - všechny buňky získají v konečném čase stejnou hodnotu 0 (např. pravidlo 40) nebo hodnotu 1.
- **CA2** - počáteční aktivita se postupně utlumuje a začínají převládat stabilní shluky (např. pravidlo 228) nebo poměrně jednoduché cyklicky se opakující struktury (pravidlo 109, obrázek 3.13).
- **CA3** - převládá zdánlivě chaotický vývin, pouhým okem není možno pozorovat pravidelné obrazce, obrazce působí jako náhodný šum (pravidlo 22, obrázek 3.12).
- **CA4** - obrazce vykazují složitou, ale přesto zřejmou pravidelnost, generují se nové struktury, které se posouvají (např. kluzáky v LIFE), struktury přetrvávají poměrně dlouho (pravidlo 110 obrázek 3.14), celulární automaty s těmito pravidly jsou schopny realizovat univerzální počítač [8].

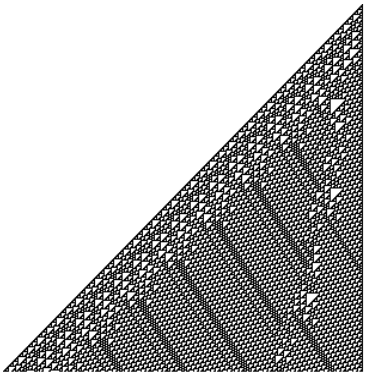
Přestože ukázka všech pravidel výše obsahuje také zde uvedená pravidla, pro lepší názornost obrázky 3.12 - 3.14 ukazují zajímavá pravidla demonstrující Wolframovy třídy.



Obrázek 3.12: Pravidlo 22 [22].



Obrázek 3.13: Pravidlo 109 [22].



Obrázek 3.14: Pravidlo 110 [22].

3.5 Kvantitativní hodnocení dynamiky CA

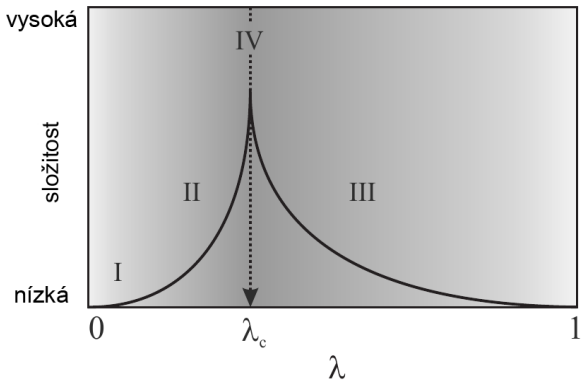
V předchozí kapitole byly definovány Wolframovy třídy, ale nebylo řečeno, jakým způsobem zařadit daný konkrétní celulární automat do které třídy. Tento problém vyřešil Langton v roce 1986 zavedením parametru λ, který charakterizuje chování celulárního automatu bez toho, aby bylo potřeba daný automat simulovat.

Parametr λ určuje míru schopnosti přenosu a uchování informace v celulárním automatu a je definován [12]:

$$\lambda = \frac{K^N - n}{K^N} \tag{1}$$

kde:

- N je počet sousedů (včetně samotné buňky),
- K je počet možných stavů buňky a
- n je počet pravidel buňky, které vedou ke klidovému stavu.



Obrázek 3.15: Závislost míry schopnosti přenosu informace na parametru λ a zároveň zobrazení rozložení Wolframových tříd [3].

Parametr λ vyjadřuje poměr počtu pravidel, jejichž výstupem jsou neklidové stavy k celkovému počtu pravidel. Klidový stav může být zvolen libovolně jako jeden z množiny stavů automatu.

Pro celulární automat s více stavy a pravidly Langton zjistil závislost, kterou vyjadřuje obrázek 3.15. Pro nízké hodnoty λ je informace v automatu „zmrazená" a nepřenáší se, tedy třída 1 a 2, pro $\lambda \approx 0.3$ je přenos možný, ale ne tak rychlý, aby se ztrácela vazba na její původní místo, jedná se o hranici mezi chaosem a řádem, tedy třída 4 a pro $\lambda \to 1$ se informace přenáší lehce až chaoticky, tedy třída 3.

Další parametr pro hodnocení dynamiky celulárních automatů je pravděpodobnostní parametr Z [24], který je definován v citované literatuře [25].

Tabulka 3.1: Vztah mezi parametrem λ, Z a Wolframovými třídami [24], kde K je počet hodnot, kterých může buňka CA nabývat.

třída	1	→	2	→	4	→	3
Z	0	...	...	...	...	...	1
λ	0	...	...	...	...	...	1-(1/k)
λ_r	0	...	...	...	...	...	1

Předchozí tabulka 3.1 znázorňuje závislost parametru λ a parametru Z na Wolframových třídách. Z tabulky je patrný konflikt s výše uvedenou definicí parametru λ v intervalu hodnot, kterých může λ nabývat. Důvodem je jiná interpretace výpočtu λ, kdy se uvažuje pouze poměr aktivních přechodů v lokální přechodové funkci vůči polovině všech možných přechodů. Do tabulky je dále zahrnut parametr λ_r [24], který je odvozen od λ následovně:

$$\lambda_r = 2\lambda \text{ pro } \lambda \leq \frac{1}{2}$$

$$\lambda_r = 2(1-\lambda) \text{ pro } \lambda > \frac{1}{2}$$

Tyto hodnotící parametry je možno použít bohužel u automatů s větším počtem pravidel, mezi které binární 1D CA s okolím o velikosti 3 nepatří. V kapitole 7 je v tabulce vypsáno všech 256 pravidel včetně λ a Z parametru a je patrné že určení Wolframovy třídy podle nich zde nefunguje.

Kapitola 4

Globálně řízené CA

4.1 Úvod do globálně řízených celulárních automatů

Předchozí kapitoly popsaly celulární automat od obecného přes jednorozměrný až po jeho klasifikaci. Globálně řízeným celulárním automatem se rozumí takový automat, ve kterém se pravidla všech buněk mohou globálně během simulace změnit. První informace o takto upraveném řízení celulárního automatu byla uvedena v článku Global Control In Polymorphic Cellular Automata [13]. Důvodem k zavedení globálního řízení celulárních automatů byla rychlejší sebereplikace Bylových smyček.

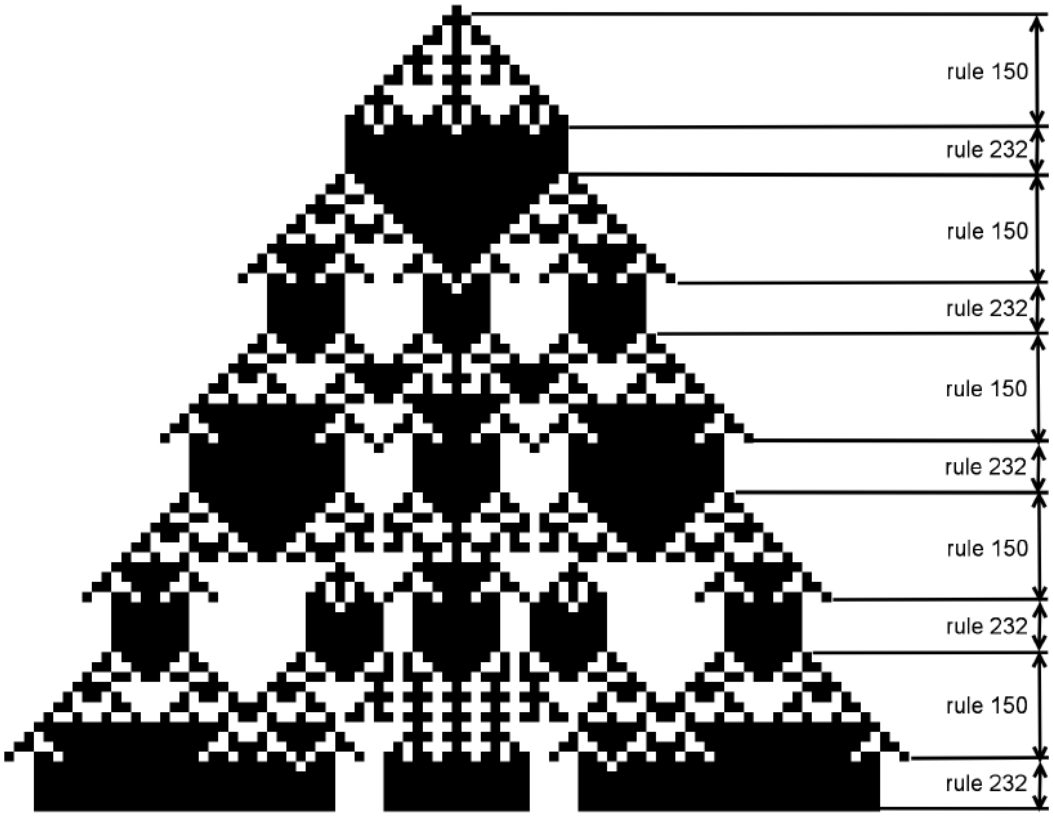
Pro další popis je uvažován 1D celulární automat s pravidly definovanými číslem v rozsahu $0 - 255$. Globálně řízený CA je na rozdíl od klasického CA definován množinou pravidel, které se mohou v buňce střídat a počtem jejich opakování. Pravidla se střídají ve všech buňkách najednou a všem buňkám je přiřazeno stejné nové pravidlo. Simulace CA probíhá z počáteční konfigurace výběrem prvního pravidla a příslušným počtem aplikací a pokračuje výběrem dalšího pravidla atd. Pravidla se z množiny pravidel vybírají cyklicky.

Obrázek 4.1 ukazuje použití 1D CA s globálním řízením a s 11 kroky pravidla 150 a s 5 kroky pravidla 232.

4.2 Formální definice

Na základě již uvedených definic získáme definici globálně řízeného celulárního automatu:

Definice 4.2.1 *Jednorozměrný binární uniformní* ***globálně*** *řízený celulární automat s konečným počtem buněk je osmice* $A = (Q, N, R, \Delta, e, b_1, b_2, c_0)$*, kde*
$Q = \{0,1\}$ *je binární množina stavů,*
N *určuje okolí* ($N \subseteq \mathbb{Z}$),
Δ *je konečná množina lokálních přechodových funkcí* $\delta_1 \dots \delta_k$, $\delta_i : Q^N \to Q$
z *určuje počet buněk,*
b_1 *a* b_2 *jsou okrajové hodnoty,*
c_0 *je počáteční konfigurace, a*
e *je funkce,* $e : \mathbb{N}^+ \to \mathbb{N}^{0+}$*, která přiřazuje každé lokální přechodové funkci z* Δ *konstantu, určující počet opakování lokální přechodové funkce*



Obrázek 4.1: Ukázka vývoje 1D CA s globálním řízením [13]

Globálně řízený celulární automat, jehož průběh je zobrazen na obrázku 4.1 by v souladu s definicí 4.2.1 byl definován jako: $A = (Q, N, R, \Delta, e, b_1, b_2, c_0)$ a kde
$Q = \{0, 1\}$
$N = \{-1, 0, 1\}$
$\Delta = \{150, 232\}$ (vyjádřeno číslem pravidla)
$z = 87$
$b_1 = 0$, $b_2 = 0$
$c_0 = \{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,$
$0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,$
$0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\}$
$e(1) = 11$
$e(2) = 5$

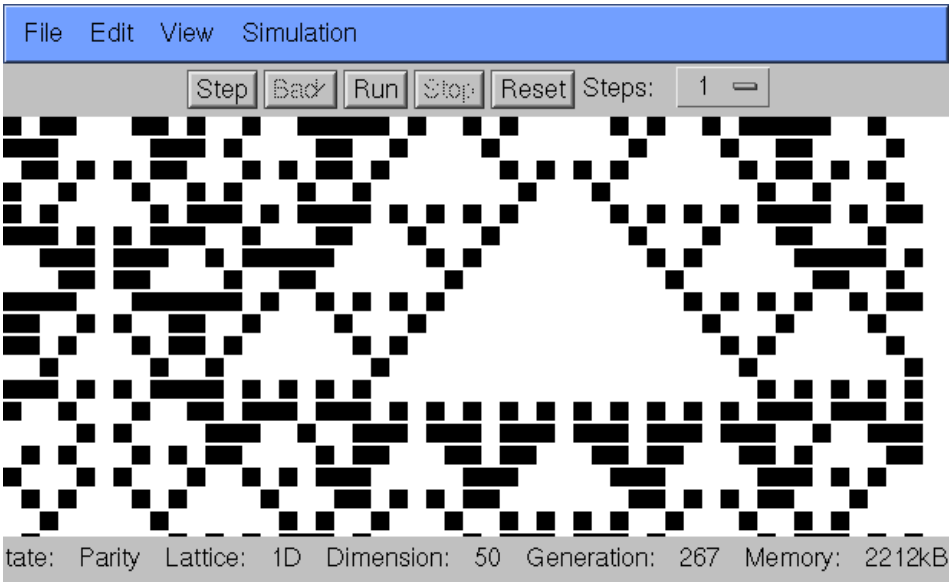
Kapitola 5

Dostupné systémy pro simulaci CA

Tato kapitola ve zkratce analyzuje dostupné simulační nástroje celulárních automatů. Cílem je nalézt takový simulační nástroj, který by snadno umožňoval především globální řízení definované například textovým vstupem a vhodný výstup umožňující analyzovat nastavené chování celulárního automatu.

5.1 JCASim

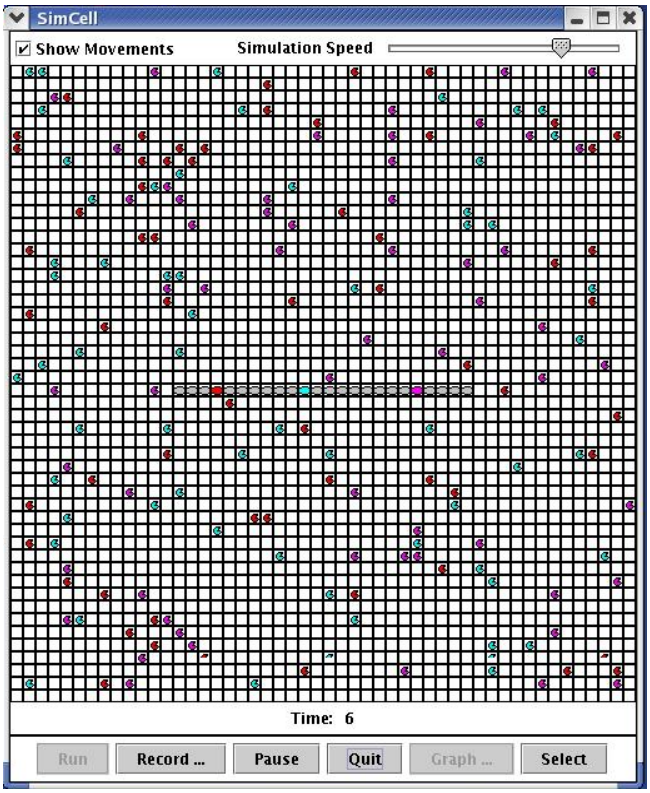
JCASim [17] je systém pro simulování celulárních automatů napsaný v jazyce Java. Je možno je spustit jako webový applet nebo standalone aplikaci. Celulární automaty mohou být specifikovány v Javě, CDL nebo přes interaktivní rozhraní. Systém podporuje mnoho různých geometrických typů mřížek (1D, 2D - čtverec, šestiúhelník, trojúhelník, 3D), různé druhy okolí a výsledek může být zobrazen formou barev, textu nebo ikon. Na obrázku 5.1 je ukázka grafického prostředí.



Obrázek 5.1: JCASim

5.2 SimCell

SimCell [16] je simulátor celulárních automatů užívaný k simulaci biochemických procesů a pro výpočet slouží DCA algoritmus. Pomocí tohoto programu je možno vytvořit: malé molekuly, membrány, proteinové membrány, RNA molekuly, DNA molekuly,... Obrázek 5.2 ukazuje grafické prostředí programu.



Obrázek 5.2: SimCell

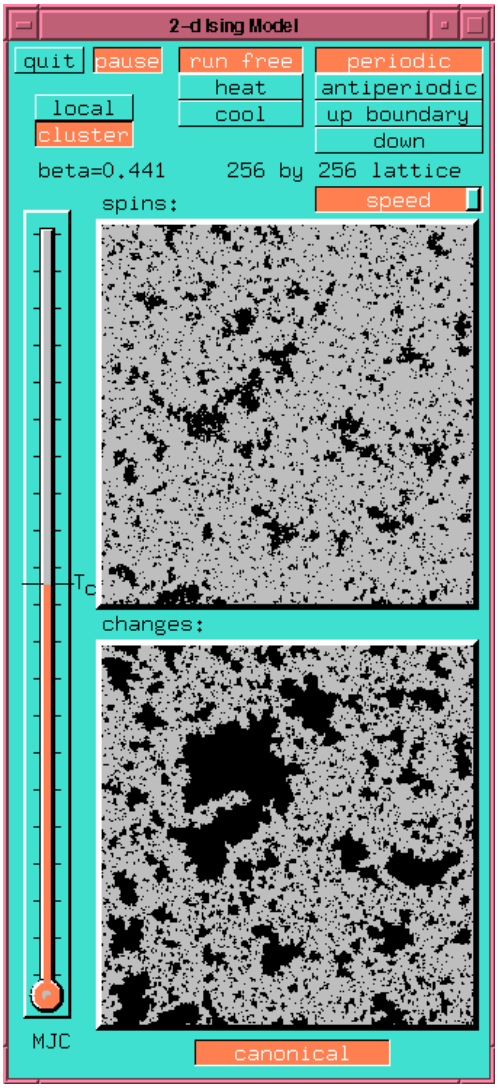
5.3 CAGE

CAGE [15] je obecný a komplexní simulátor celulárních automatů napsaný v jazyce Python. Podporuje 1D a 2D automaty, řadu před-připravených pravidel, koncept agentů, kteří se mohou pohybovat nezávisle v prostoru podle implementovaného chování.

CAGE obsahuje mnoho plně funkčních příkladů, včetně Game of Life, Langtonových sebe-reprodukujích se smyček a 1D automatů. Systém obsahuje také jednoduché grafické prostředí sloužící k zobrazení výsledků. CAGE je určen spíše ke vzdělávacím účelům než jako výkonný simulátor.

5.4 Xtyos

Xtoys [1] je simulační systém napsaný v jazyce C pro X-window prostředí. Obsahuje několik demonstračních příkladů. Grafické rozhraní je zobrazeno na obrázku 5.3.



Obrázek 5.3: Xtoys

5.5 Vyhodnocení

Analyzované simulační nástroje, které byly v této kapitole uvedeny, nezahrnují jistě všechny dostupné simulační nástroje, ale jsou ukázkou dostupných prostředků. Klasifikace globálně řízených celulárních automatů, která je uvedena v následujích kapitolách, je hlavním cílem této práce, a proto bylo věnováno úsilí vytvoření vlastního simulátoru, místo podrobné analýzy uvedených nástorojů. Hlavní důvod k tomuto rozhodnutí byla absence možnosti globálního řízení, jakožto nativní vlastnosti simulovaných automatů.

Kapitola 6

Implementace simulátoru

Tato kapitola jako celek se zabývá implementací vlastního simulátoru globálně řízených celulárních automatů. Nejprve je uvedena specifikace požadavků na simulátor, poté popsán jazyk Java a tvorba grafického uživatelského rozhraní v tomto jazyce. Dále je popsán vlastní simulátor, nejdříve z pohledu uživatele, tedy funkcionality a následně z pohledu programátora.

6.1 Specifikace

Pro provedení experimetů s globálně řízenými celulárními automaty bylo potřeba vytvořit platformě nezávislý simulátor globálně řízených celulárních automatů v souladu s dříve uvedenou definicí 4.2.1. Simulátor musí umožnit grafické zobrazení výpočtu (sekvence konfigurací) automatu. Definice automatu musí být oddělena od simulátoru - definovat vhodný formát a načítání definice ze souboru, případně umožnit export simulace. Grafické uživatelské rozhraní musí simulaci pouze zobrazovat, nesmí být s ní funkčně spjato.

Na základě uvedených požadavků byl zvolen implementační jazyk Java s grafickým prostředím Swing, především pro jednoduchou přenositelnost. Proto jsou následující dvě kapitoly věnovány tomuto jazyku a možnosti implementace grafického uživatelského rozhraní.

6.2 Java

Java je objektově orientovaný programovací jazyk vyvinutý firmou Sun Microsystems (nyní součást Oracle Corporation) [14], který byl poprvé představen 23. května 1995 jako základní součást platformy Java. Syntaxe jazyka byla odvozena od jazyků C a C++[19], ale byl použit jednodušší objektový model a méně nízko úrovňových konstrukcí. Aplikace napsané v jazyce Java jsou kompilovány do byte kódu (class soubory), které jsou interpretovány Java Virtual Machine (JVM), nezávisle na fyzické architektuře. Java je určená pro vývojáře, kteří chtějí své aplikace psát jednou a spouštět kdekoli, formálněji řečeno, Java Virtual Machine je dostupná pro mnoho architektur, počínaje čipovými kartami (platforma JavaCard), přes mobilní a přenosná zařízení (Java Micro Edition), aplikace pro klasická PC (platforma Java Standard Edition) až po rozsáhlé distribuované systémy (platforma Java Enterprise Edition). V roce 2007 Sun uvolnil zdrojové kódy Javy pod licencí GNU General Public License a vývoj bude nadále probíhat jako open source. Běhové prostředí pro Standard edition (Java runtime enviroment JRE) je v současné době dostupné pro následující operační

systémy: Microsoft Windows (7/XP/Vista/2000/2003/2008), Solaris (32-bit, 64-bit), Linux (32-bit, 64-bit), Apple (OS X) má vlastní implementaci JRE.

Java používá ke správě paměti automatický garbage collector, který se stará o přidělenou paměť po celou dobu života objektu. Programátor určuje, kdy bude objekt vytvořen a Java runtime je zodpovědný za uvolnění paměti, pokud se objekt již nepoužívá. Pokud již neexistuje žádná reference na daný objekt, paměť je označena k uvolnění pro garbage collector. K situaci podobné úniku paměti (memory leak) může dojít, pokud kód programu udržuje již nadále nepotřebné objekty v ještě používaných kontejnerech. Pokud dojde k volání metody neexistujícího objektu, dojde k vyvolání výjimky "null pointer exception". K provedení garbage collectoru může dojít kdykoli, v ideálním případě pokud je program nečinný. Pokud dojde volná paměť, provedení garbage collectoru je zaručeno, což může způsobit zpomalení odezvy programu. Explicitní správa paměti v Javě není možná.

Java nepodporuje ukazatelovou aritmetiku známou z C / C++, kdy adresa objektu je reprezentována neznaménkovým dlouhým celým číslem (unsigned long integer). Toto řešení umožňuje garbage collectoru realokovat objekty a zajišťuje typovou kontrolu a bezpečnost. Stejně jako v C++ a některých jiných objektově orientovaných jazycích nejsou primitivní datové typy reprezentovány objekty. Toto řešení bylo vědomě zvoleno vývojáři Javy z výkonnostních důvodů, a proto také není Java považována za čistě objektově orientovaný programovací jazyk.

6.3 Grafická rozhraní

V době vzniku jazyka Java nebyla podpora grafického rozhraní příliš propracována a ze strany vývojářů oblíbena. Během vývoje bylo vytvořeno několik implementací pro tvorbu grafických uživatelských rozhraní.

Knihovna AWT (Abstract Window Toolkit) [4] je první implementací grafického uživatelského rozhraní, která byla dodána společně s Javou a je dostupná od verze JRE 1.0. AWT je pouze základní knihovna pro programování grafického uživatelského rozhraní, samotná knihovna neobsahuje žádné grafické uživatelské prvky, jako jsou tlačítka, seznamy,... Tato funkcionalita je poskytována zobrazením prvků z nativních systémových knihoven operačního systému. AWT je také odpovědno za obsluhu vstupních uživatelských událostí, jako je stisk klávesy nebo kliknutí myši. Události, které se vyskytnou v nativním systémovém okně, obdrží AWT, které je předá Java aplikaci jako AWT události. Přestože se jedná o abstraktní (platformě nezávislé) rozhraní, přenositelnost je problematická vzhledem k rozdílným vlastnostem nativní úrovně uživatelských prvků.

V současné době je AWT nadále funkční a je možno jej využívat tak, jak bylo v době vzniku předpokládáno. Zpětná kompatibilita je zaručena, i když vývoj postoupil a většina desktopových aplikací využívá knihovny Swing, protože Swing poskytuje mnohem flexibilnější a silnější nástroje pro tvorbu grafického uživatelského rozhraní. Přesto AWT zastává kritickou část uživatelského rozhraní tím, že zajišťuje významné platformě závislé funkce, na kterých závisí i Swing. Například aplikační okna ve Swingu využívají Swing komponentu JFrame, která využívá AWT pro vytvoření aktuálního okna na obrazovce. Knihovna AWT také poskytuje některé základní funkce, na kterých závisí Swing, jako jsou mechanismus událostí, funkce vyjmout&vložit, funkce drap&drop a správa vstupních činností uživatele.

Na schématu 6.1 je vidět závislost jednotlivých implementací. Komponenty z knihovny AWT je možno použít přímo nebo nepřímo prostřednictvím Swing API, které na AWT závisí.

Tabulka 6.1: Hierarchie API grafických desktopových Java aplikací [4].

<table>
<tr><td colspan="3">Java aplikace</td></tr>
<tr><td></td><td>Swing</td><td></td></tr>
<tr><td colspan="2">AWT</td><td>Java 2D</td></tr>
<tr><td colspan="3">Java Runtime Enviroment</td></tr>
</table>

Java 2D je grafická knihovna Javy [4] poprvé představená v JDK (Java Developer Kit) 1.2. Přestože AWT obsahuje základní API pro kreslení od verze JDK 1.0, Java 2D jde mnohem dále a zahrnuje širokou škálu operací, včetně základních a pokročilých technik kreslení, manipulace s obrázky, textem a tiskem. Funkcionalita Java 2D je využita ve Swingových komponentech, např. když má být na obrazovce zobrazeno tlačítko, Swing volá Java 2D a vykreslí pozadí, ohraničení a text tlačítka.

Swing je knihovna uživatelských prvků na platformě Java pro ovládání počítače pomocí grafického rozhraní. Knihovna Swing poskytuje aplikační rozhraní pro tvorbu a obsluhu klasického grafického uživatelského rozhraní [18]. Swing, stejně jako Java 2D byl představen v JDK 1.2. Knihovna Swing je nejčastěji využívána dnešními vývojáři desktopových aplikací s grafickým rozhraním v Javě. Swing komponenty jsou lehké, to znamená, že Swing komponenty, které jsou zobrazeny na obrazovce, neodpovídají původním (systémovým) komponentám, jako je tomu v AWT (tyto systémově vázané komponenty se nazývají těžké). Tento rozdíl je zcela irelevantní pro koncového uživatele, pokud objekt vypadá jako komponenta, lze na něj kliknout jako na komponentu a reaguje jako komponenta, potom je komponenta.

Značný rozdíl v implementaci AWT a Swingu je pro vývojáře, protože Swing komponenty jsou vykreslovány pomocí Java 2D a tím je umožněno jejich přizpůsobení pro konkrétní potřeby programu, což umožňuje aplikacím vypadat a chovat se zajímavěji. Swing je v základních operacích, jako je např. vytvoření nového okna, vázán na AWT, viz schéma 6.1.

Na obrázku 6.1 je podrobně zobrazena hierarchie tříd Swingu dle JDK 1.4. Na první pohled je patrná podobnost s hierarchií v AWT. Každá Swing komponenta s odpovídající komponentou v AWT sdílí stejný název s tím rozdílem, že název Swing komponenty má prefix "J".

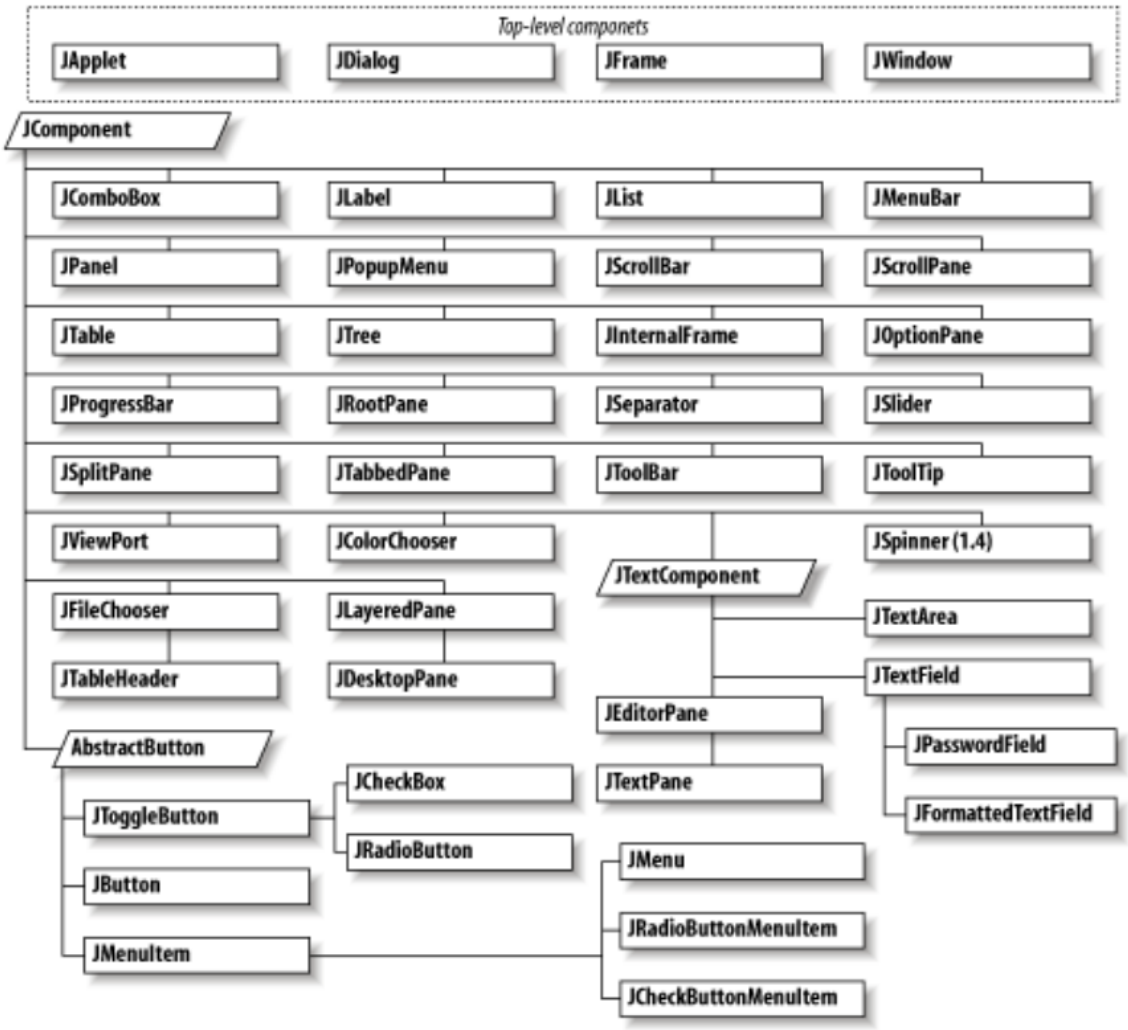
Horní oblast obrázku 6.1 zobrazuje komponenty s nejvyšší úrovní, kterými jsou `JApplet` pro webové applety, `JDialog` pro desktopová dialogová okna, `JFrame` a `JWindow` pro desktopová okna aplikací.

Zbytek obrázku jsou třídy dědící od třídy `JComponent`, tedy veškeré uživatelské prvky, které lze umístit do okna aplikace. Například aplikační menu, rozbalovací nabídky, popisky, scrolovací oblasti, dialogy pro výběr souboru, barvy, textové oblasti, tlačítka atd.

Tato implementace nese s sebou určité výhody, jako jsou: kompletní implementace v Javě, vše je tedy platformě nezávislé; standardní implementace lze nahrazovat vlastními a měnit tak chování komponent; důsledné oddělení funkcionality od vzhledu (look&feel),

témat vzhledu je několik dostupných přímo ve standardní knihovně, další lze vytvořit a připojit; u složitějších aplikací oddělení dat od grafického uživatelského rozhraní [5].

Uvedené vlastnosti jsou pro programátory velmi přínosné, ale jsou vyváženy relativně vysokými paměťovými nároky a nároky na procesor. V reakci na tuto skutečnost vznikly grafické knihovny, které se snaží odstranit tento problém. Nejvýznamnější takováto knihovna je SWT.



Obrázek 6.1: Hierarchie komponent Swingu [10]

Standard Widget Toolkit (SWT) je další knihovna grafických uživatelských prvků pro platformu Java [20]. Původ této knihovny je ve firmě IBM a nyní je ve správě nadace Eclipse Foundation, spolu s projektem Eclipse IDE. SWT je alternativou k AWT v tom smyslu, že také primárně využívá nativní komponenty operačního systému k vykreslování uživatelských prvků prostřednictvím JNI (Java Native Interface). Programy, které využívají SWT, jsou přenosné, ale implementace SWT, ačkoli je napsána v Javě, je unikátní pro každou platformu.

SWT je obálkou okolo nativních systémových objektů - GTK+ objektů, Win32 objektů atd. Z tohoto důvodu jsou SWT komponenty označeny jako "těžké". V případě, kdy nativní objekty neposkytují požadovanou funkcionalitu, SWT má vlastní implementaci, obdobně

jako ve Swingu. Lze konstatovat, že SWT je jakýmsi kompromisem mezi vzhledem a nízkým výkonem z AWT a vysokou úrovní používání ve Swingu.

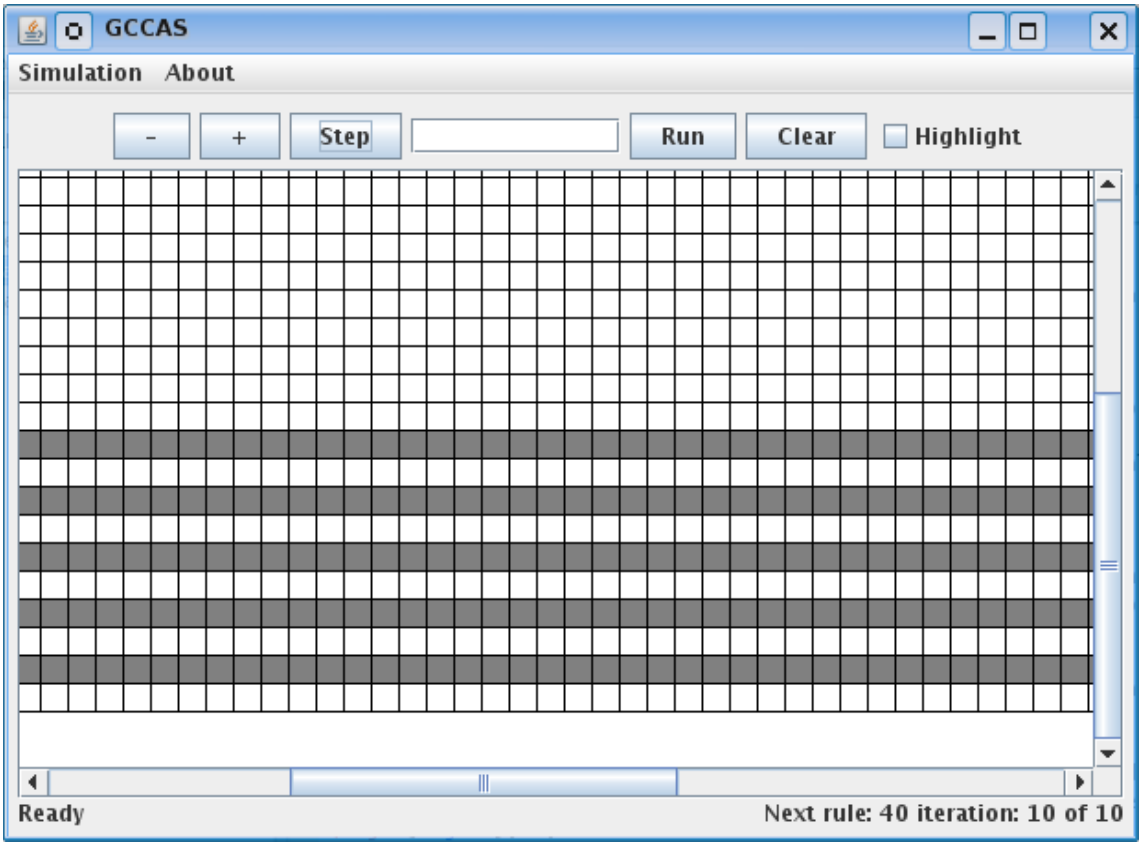
6.4 Implementace GCCAS

Pro vytvoření GCCAS - Globally Controlled Cellular Automata Simulator byl zvolen jazyk Java a grafické prostředí Swing, pro vývoj bylo zvoleno integrované vývojové prostředí Eclipse, vývojové verze byly průběžně ukládány na SVN.

Následující kapitoly popisují vytvořený simulátor z pohledu koncového uživatele, dále popisují možnosti a požadavky programu.

6.4.1 Uživatelské rozhraní

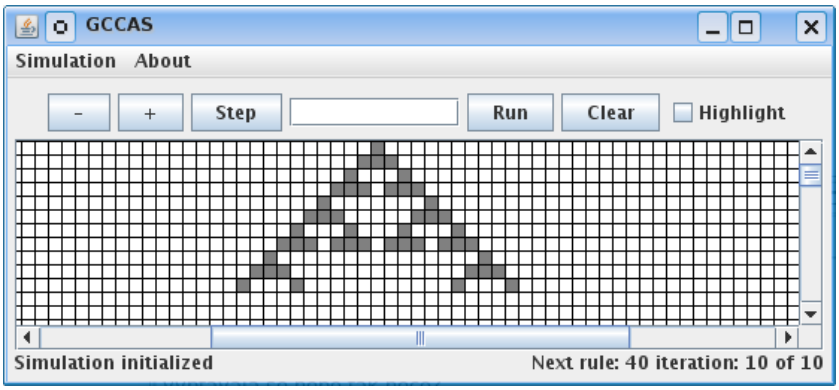
Uživatelské rozhraní simulátoru bylo navrženo s ohledem na předpokládané využití, tedy provedení experimentálních testů. Od toho byly odvozeny a implementovány funkce programu.



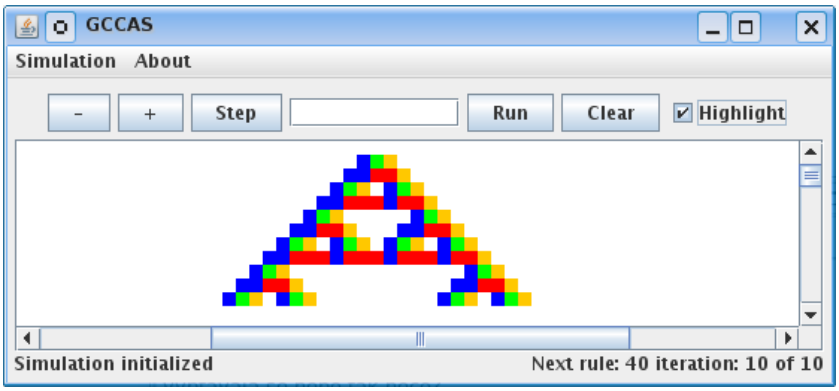
Obrázek 6.2: Okno simulátoru GCCAS.

Z hlavního menu Simulation je možno simulaci načíst ze souboru, exportovat a program ukončit. Tlačítky "+"a "-"lze zvětšit, resp. zmenšit velikost buňky. Tlačítkem Step se provede jeden krok výpočtu. Do textového pole uprostřed je možno zadat počet požadovaných kroků výpočtu a tlačítkem Run je provést najednou. Tlačítko Clear znovu inicializuje simulaci, ponechá tedy pouze první konfiguraci. Ve stavové liště v dolní části okna aplikace je

zobrazena informace o pravidle, které bude použito při příštím výpočtu a počtu opakování. Zaškrtávací políčko Highlight přepíná režim zobrazení simulace z klasického do barevného. V klasickém zobrazení šedé políčko označuje buňku s hodnotou 1 a bílé políčko buňku s hodnotou 0. V barevném zobrazení jsou zvýrazněny buňky, které byly ovlivněny aktivní (s hodnotou 1) buňkou v předchozím kroku [25]. Toto zobrazení by mělo usnadnit vizuální klasifikaci automatů. Rozdíl zobrazení je patrný z obrázků 6.3 a 6.4.



Obrázek 6.3: Zvýraznění vypnuto.



Obrázek 6.4: Zvýraznění zapnuto.

Vytvořená aplikace je v `jar` archivu a spuštění se provede z příkazové řádky příkazem: `java -jar gccas.jar` v adresáři, ve kterém je uložen simulátor.

6.4.2 Konfigurace simulace

Simulaci se konfiguruje v externím textovém souboru, není tedy možné parametry automatu nastavit přes grafické rozhraní. Konfigurační soubor do jisté míry kopíruje definici globálně řízeného celulárního automatu. Struktura konfiguračního souboru je klíčové slovo následované parametry.

Klíčová slova konfiguračního souboru:
`initialize values` *hodnota* - počáteční konfigurace, odpovídá složce c_0, hodnotu buněk je možno zadat buď binárně postupně po jednotlivých buňkách, nebo kódem ve tvaru *počet_opakování hodnota_buňky počet_opakování hodnota_buňky* ...,

`rule` *pravidlo počet_opakování* - definuje, které pravidlo a kolikrát bude použito při výpočtu automatu, odpovídá složkám Δ a e, příkaz s tímto klíčovým slovem musí být uveden minimálně jednou, maximální počet pravidel není explicitně omezen,

`boundary` *typ hodnota* - definuje okrajové podmínky, odpovídá složkám b_1 a b_2, jsou k dispozici dva typy okrajových podmínek a sice `periodic` - periodické a `fixed` *hodnota* - pevné okrajové podmníky, parametr *hodnota* určuje hodnotu virtuálních buněk z množiny {0,1},

`run` *počet_kroků* - volitelný parametr, který, pokud je uveden v konfiguračním souboru, definuje počet výpočtů simulace a automatické spuštění simulace po načtení souboru,

`//` - řádek začínající dvěmi lomítky je považován za komentář a není simulátorem zpracován.

Ukázka konfiguračního souboru s počáteční konfigurací po jednotlivých buňkách, dvěma pravidly a periodickými podmínkami je na obrázku 6.5. Druhá ukázka na obrázku 6.6 využívá definici počáteční konfigurace pomocí počtu buněk a hodnoty, dále má 3 pravidla, pevné okrajové podmínky a po načtení konfiguračního souboru se spustí výpočet 1000 kroků.

```
initialize values 01010010011101010100110101010101011101
rule 2 15
rule 040 10
boundary periodic
```

Obrázek 6.5: Ukázka konfiguračního souboru simulátoru GCCAS.

```
initialize code 50 0 3 1 5 0 2 1 50 0
rule 2 15
rule 040 10
rule 128 10
boundary fixed 0
run 1000
```

Obrázek 6.6: Ukázka konfiguračního souboru simulátoru GCCAS.

6.4.3 Struktura programu

Návrh struktury programu byl navržen s ohledem na specifikaci, především s cílem oddělit výpočet simulace od grafického zobrazení. Třídy jsou rozděleny do tří balíčků: `gui`, `kernel` a `tools`.

Balíček gui

Tento balíček obsahuje třídy, které obstarávají grafické zobrazení simulace na obrazovce, obsahuje také hlavní třídu pro spuštění programu. Jednotlivé třídy:

`ErrorHandler` - objekt této třídy je dostupný z hlavního okna aplikace a zajišťuje vypsání chybových, resp. oznamovacích zpráv uživateli, standardní výpis probíhá do stavového řádku aplikace,

`Gui` - hlavní třída definující rozložení grafických prvků v okně aplikace, dále definuje reakce na vstupní události a obsahuje metodu `main`, která se volá při spuštění aplikace,

`RuleLabel` - potomek Swing třídy JLabel, zajišťuje vypsání aktuálního pravidla a počtu opakování ve stavovém řádku programu,

`SimulatorPanel` - potomek Swing třídy JPanel, metoda `void paint(Graphics g)` je předefinována a je z ní volána metoda `void paintSimulation(Graphics g)`, ve které je vykreslen aktuální stav simulace, data pro vykreslení jsou získána z instance objektu `Simulator`, vykreslení probíhá v cyklu po řádcích, barva vykreslených čtverců (buněk) je řízena testem na zapnuté zvýraznění, metoda `void checkSize()` nastavuje grafickou velikost objektu `SimulatorPanel` v závislosti na počtu buněk v řádku a sloupci, pokud je velikost větší než aktuální velikost okna, jsou zobrazeny posuvníky.

Balíček kernel

V tomto balíčku jsou třídy pro řízení a výpočet simulace a pro načtení konfiguračního souboru.

`Boundary` - tato třída reprezentuje okrajové podmínky, obsahuje metody
`CACell getLcell(Step step)` a `CACell getRcell(Step step)`, které vrací na základě posledního výpočtu a konfigurace simulace levou, resp. pravou chybějící buňku, implementovány jsou pouze dvě výše uvedené varianty, ale rozšíření o další typy znamená pouze doimplementovat výpočet v této třídě,

`CACell` - třída reprezentující jednu buňku celulárního automatu, obsahuje informaci o své hodnotě a o hodnotě v režimu zvýraznění,

`Command` - obdoba složek Δ a e z definice globálně řízeného celulárního automatu, tato třída obsahuje pravidlo pro výpočet a počet opakování,

`InputFile` - tato třída zajišťuje správné načtení konfiguračního souboru a převedení textových informací na objekty simulace, které jsou přes metody objektu dostupné simulátoru,

`Rule` - pravidlo pro výpočet simulace je v simulátoru reprezentováno touto třídou, dekadická hodnota pravidla je převedena na přechodovou tabulku a metoda
`CACell getNewValue(CACell lCell, CACell cell, CACell rCell)` při zadání okolních buněk z posledního kroku vrací buňku pro aktuálně vypočítávaný krok[1],

`Simulation` - tato třída je rozšířením třídy `Vector`, obsahuje Vector kroků, jedná se tedy o reprezentaci matice buněk, tedy o všechny výpočty aktuální simulace,

[1] V případě reprezentace okolí buňky byl objektový model aplikace zjednodušen, je pevně uvažováno okolí o velikosti tří buňek. Toto zjednodušení bylo provedeno, protože nebylo uvažováno provádění simulace s jiným nastavením, resp. s jinými automaty, než je třída základních binárních jednorozměrných celulárních automatů.

`Simulator` - pro tuto třídu byl využit návrhový vzor Singleton, instance této třídy je v rámci jednoho spuštění aplikace pouze jedna a je jádrem řízení simulace, obsahuje metody pro iniciaizace simulace
`void initialize(Step initValues, Vector<Command> commands, Boundary boundary)`, pro výpočet nového kroku `void nextStep()` a další,

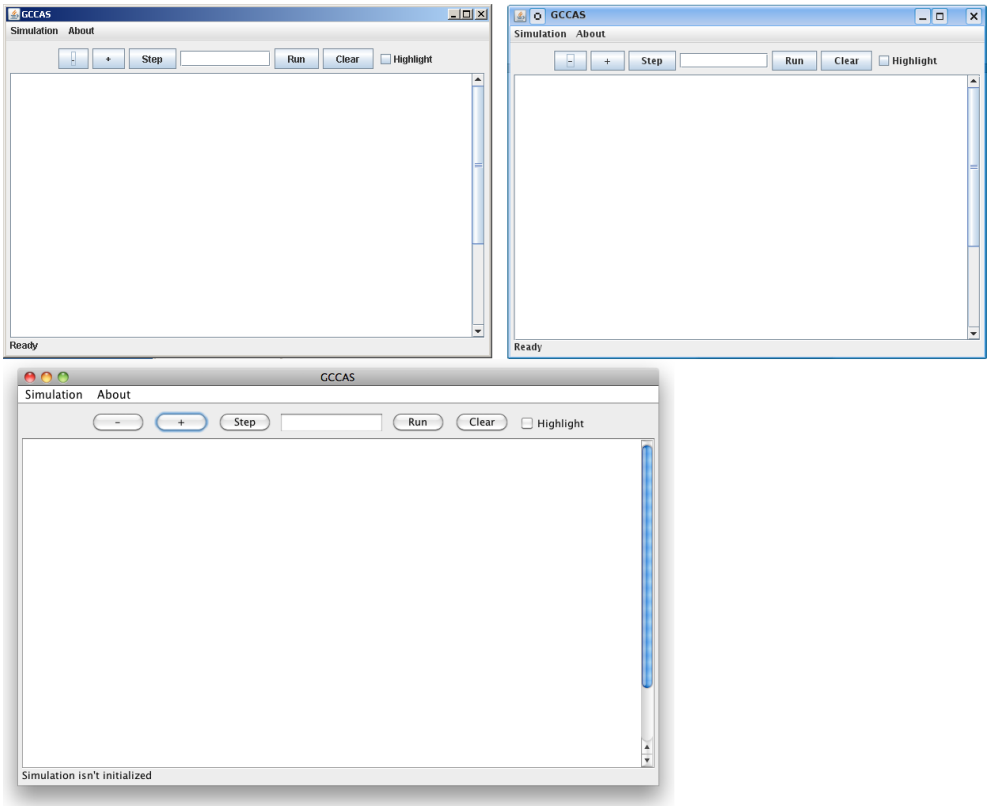
`Step` - reprezentuje jeden krok výpočtu, jedná se o Vector buněk, tedy o `Vector<CACell>`.

Balíček tools

Balíček tools je vytvořen pro třídy, které mají širší využití a nesouvisí přímo se simulací.

`ReverseString` - statická třída poskytující metodu pro přehození pořadí písmen v řetězci (reverzaci).

Obrázek 6.7 je potvrzením výše uvedených tvrzení o přenositelnosti aplikací v jazyce Java a grafického rozhraní Swing, zobrazuje aplikaci simulátoru na třech různých platformách.



Obrázek 6.7: Ukázka přenositelnosti aplikací v jazyce Java. Aplikace GCASS na (zleva) Windows 7, Linux, Mac OS. Na všech platformách funkční a se zachováním uživatelského rozhraní.

Kapitola 7

Klasifikace pravidel celulárních automatů - Wolframovy třídy

Tato kapitola se zabývá klasifikací základních 256 pravidel jednorozměrného celulárního automatu s okolím o velikosti tří buněk (včetně buňky samotné).

Jak již bylo uvedeno v kapitole 3.5, automatická klasifikace na základě parametru λ pro uvažovaný 1D celulární automat nefunguje. Vzhledem k tomu, že ani prostudovaná literatura není v zařazení pravidel do Wolframových tříd jednotná[1], bylo zvoleno referenční ohodnocení pravidel, na základě kterého byly později provedeny experimenty a testy s hodnocením globálně řízených celulárních automatů. Toto referenční ohodnocení je uvedeno v příloze C, která obsahuje všechny prostudované a potřebné vlastnosti a zároveň je patrný nesoulad mezi parametry a Wolframovými třídami. Data pro referenční tabulku vznikla sloučením ze dvou zdrojů. Klasifikace do tříd byla převzata dle webu Wolfram Aplha [23], hodnoty parametru Z z přílohy dokumentu Generative Music and Cellular Automata [2].

V následujících tabulkách 7.1, 7.2, 7.3 a 7.4 jsou uvedena pravidla, která byla použita v experimentálních testech v kapitole 8.2.

Tabulka 7.1: Výběr pravidel třídy 1.

Pravidlo	binárně								Z	λ	třída	obživnutí
40	0	0	1	0	1	0	0	0	0,50	0,250	1	
128	1	0	0	0	0	0	0	0	0,25	0,125	1	
253	1	1	1	1	1	1	0	1	0,25	0,875	1	ano
239	1	1	1	0	1	1	1	1	0,25	0,875	1	ano

[1]Publikace Global Dynamics of Cellular Automata [24] na straně 150 (a dalších odvozených) uvádí ekvivalenci mezi pravidly 41, 97, 107, 121. Pokud jsou tedy tato čtyři pravidla ekvivalentní, musí být zařazena do stejné třídy. Ekvivalentními pravidly se rozumí, že ze stejných výchozích podmínek generují stejný, nebo stranově obrácený vzor. Web Wolfram Aplha [23] řadí pravidlo 41 do třídy 4 a pravidlo 97 do třídy 2. Je tedy patrný rozpor.

Tabulka 7.2: Výběr pravidel třídy 2.

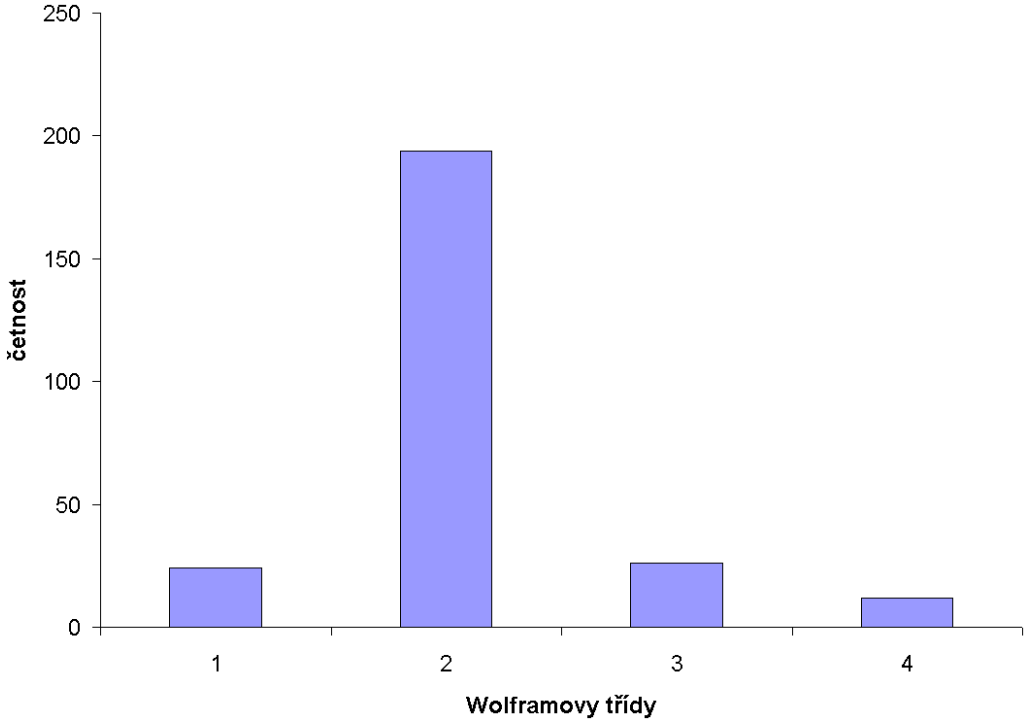
Pravidlo	binárně								Z	λ	třída	obživnutí
228	1	1	1	0	0	1	0	0	0,75	0,500	2	
109	0	1	1	0	1	1	0	1	0,75	0,625	2	ano
21	0	0	0	1	0	1	0	1	0,75	0,375	2	ano
123	0	1	1	1	1	0	1	1	0,50	0,750	2	ano

Tabulka 7.3: Výběr pravidel třídy 3.

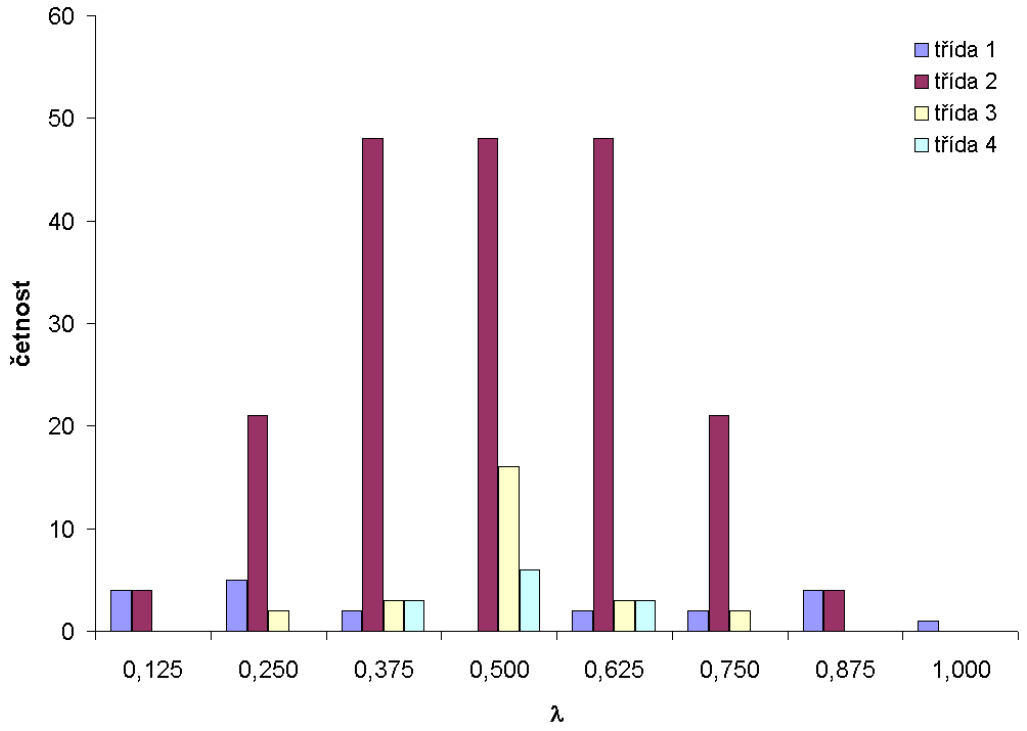
Pravidlo	binárně								Z	λ	třída	obživnutí
22	0	0	0	1	0	1	1	0	0,75	0,375	3	
45	0	0	1	0	1	1	0	1	1,00	0,500	3	ano
60	0	0	1	1	1	1	0	0	1,00	0,500	3	
129	1	0	0	0	0	0	0	1	0,50	0,250	3	ano

Tabulka 7.4: Výběr pravidel třídy 4.

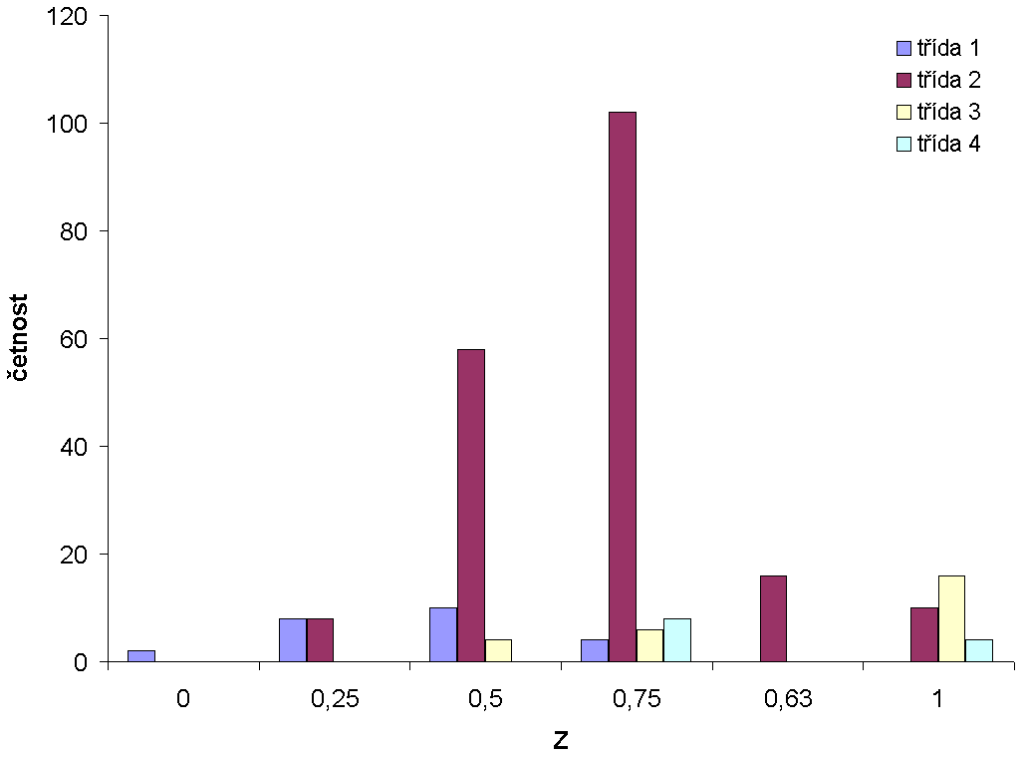
Pravidlo	binárně								Z	λ	třída	obživnutí
54	0	0	1	1	0	1	1	0	0,75	0,500	4	
110	0	1	1	0	1	1	1	0	0,75	0,625	4	
147	1	0	0	1	0	0	1	1	0,75	0,500	4	ano
137	1	0	0	0	1	0	0	1	0,75	0,375	4	ano



Obrázek 7.1: Počet pravidel v jednotlivých Wolframových třídách.



Obrázek 7.2: Počet pravidel v závislosti na parametru λ



Obrázek 7.3: Počet pravidel v závislosti na parametru Z

Všechna analyzovaná pravidla, která jsou uvedená v příloze C, byla shrnuta do grafu 7.2, který vyjadřuje závislost třídy na parametru λ a grafu 7.3, který vyjadřuje závislost třídy na parametru Z. Oba uvedené grafy by měly kopírovat rozložení tříd uvedené v tabulce 3.1 nebo rozložení zobrazené v obrázku 3.15. Z grafů je patrné, že tyto předpoklady nefungují, především pravidla z 2. třídy jsou rozložena do celého intervalu obou parametrů, přestože by měla být v omezeném intervalu a intervaly jednotlivých tříd by se neměly takto překrývat, aby bylo možno využít těchto parametrů ke klasifikaci pravidel. Graf 7.1 vyjadřuje počet pravidel v jednotlivých Wolframových třídách. Z grafu je patrné, že nejvíce pravidel je zařazeno do 2. třídy a počet pravidel v 1., 3. a 4. třídě je přibližně stejný ale zároveň řádově nižší oproti 2. třídě.

Kapitola 8

Charakterizace globálně řízených 1D celulárních automatů

Tato kapitola se věnuje cíli této diplomové práci, tedy empirické klasifikaci globálně řízených celulárních automatů do čtyř Wolframových tříd. Klasifikace musí být provedena na základě definice celulárního automatu, tedy klasifikace bez simulace.

8.1 Vizuální klasifikace

Globálně řízené automaty mohou generovat částečně odlišné struktury než klasické celulární automaty. Tento jev je způsoben přepínáním pravidel, tedy podstatou globálního řízení. Nejlépe patrný je tento rozdíl u kombinace pravidel z 1. třídy, která pokud vygenerují všechny buňky s hodnotou "0"se už nemohou z tohoto stavu dostat. Pokud je druhé pravidlo z kombinace s vlastností "obživnutí", generuje tento automat struktury z 2. třídy, tedy opakující se vzory, viz například obrázek 8.1b.

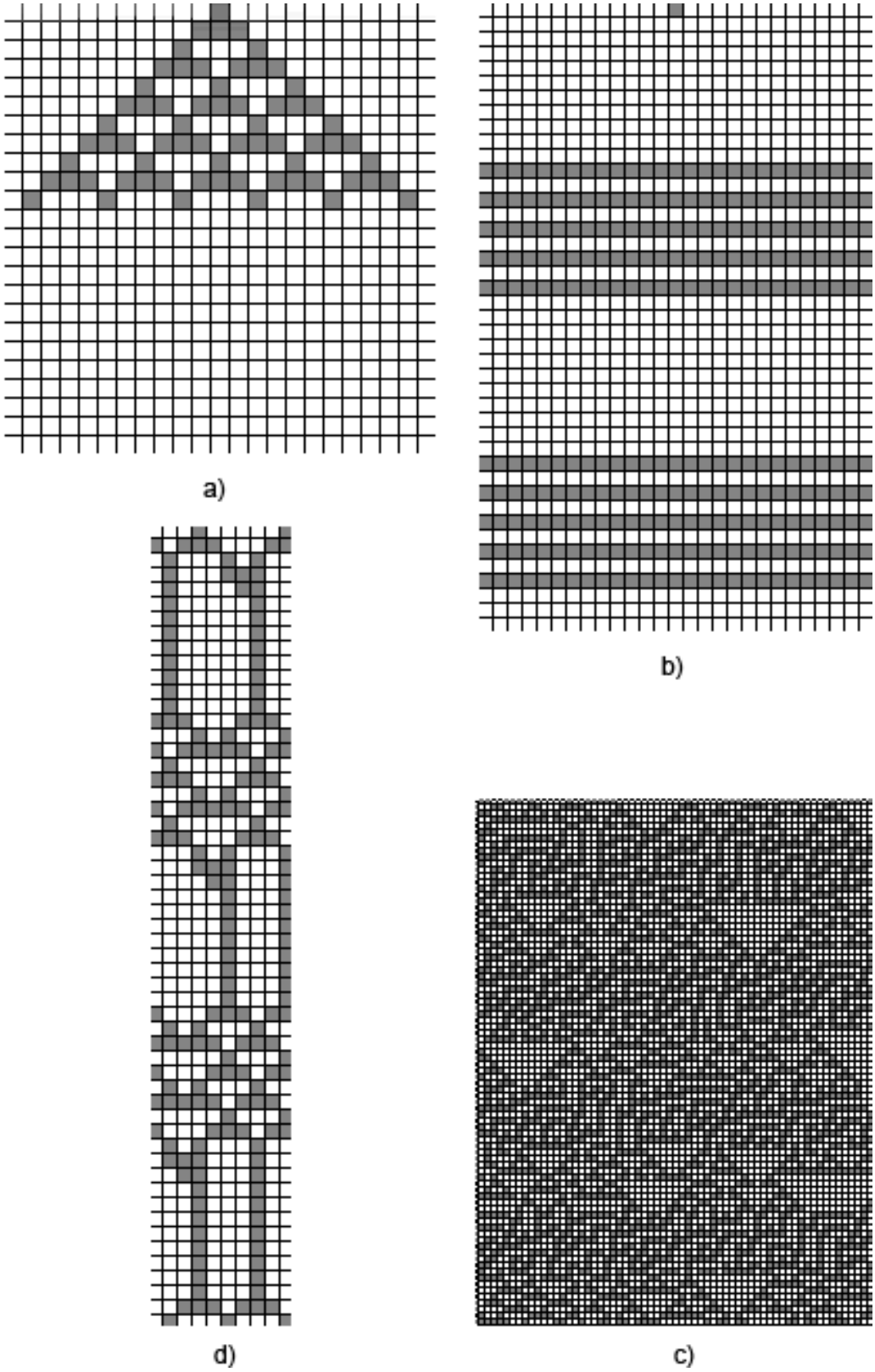
Obrázek 8.1 shrnuje vizuální vlastnosti jednotlivých Wolframových tříd pro globálně řízené celulární automaty[1]. Struktura 8.1a je klasický příklad 1. třídy, jedná se o kombinace pravidla 54 (4. třída) a 40 (1.třída), struktura generovaná pravidlem 54 je po přepnutí utlumena pravidlem 40 a tento stav je již konečný, pravidlo 54 neumí již zregenerovat. Výslednou strukturou jsou buňky s hodnotou 0.

Obrázek 8.1b je příklad 2. třídy, kombinace pravidla 128 z 1. třídy a pravidla 41 z 3. třídy. Tato struktura je charakteristická pro výše popisovanou vlastnost globálně řízených celulárních automatů, situaci kdy po přepnutí pravidla buňky zregenerují z nulového stavu. Výsledkem je struktura, která je zařazena do 2. třídy, opakující se stabilní shluky buněk.

Obrázek 8.1c je ukázka chaosu, pouhým okem nerozeznatelné pravidelné struktury, nepravidelný vzor. Tato konkrétní struktura je kombinací pravidla 45 ze 3. třídy a pravidla 22 také ze 3. třídy. Výsledná struktura zůstává ve 3. třídě.

Na obrázku 8.1d je patrná struktura, která se v čase opakuje a v prostoru se posouvá směrem doleva. Toto je typický příklad 4. třídy, v tomto případě struktura vznikla kombinací pravidla 228 z 2. třídy a pravidla 54 ze 4. třídy.

[1]Předpokládá se, že způsob klasifikace do Wolframových tříd uvedený u celulárních automatů lze použít také pro automaty s globálním řízením.



Obrázek 8.1: Ukázka vizuální klasifikace automatů do Wolframových tříd, a) 1. třída, b) 2. třída, c). 3. třída, d) 4. třída.

8.2 Experimentální testy

V této diplomové práci byl navržen postup pro klasifikaci globálně řízených celulárních automatů, který se zakládá na experimentálním ohodnocení vhodně zvolenýcch kombinací pravidel různých tříd. Předpokládá se, že poznatky získané na zvoleném vzorku pravidel budou aplikovatelné také pro ostatní pravidla. Z každé třídy byla vybrána čtyři pravidla, která danou třídu vhodně reprezentují a jsou uvedena v kapitole 7. Pravidla byla dále rozdělena na dvě skupiny po dvou a byla vytvořena matice prvních dvojic, která vytvořila kombinace "každý s každým". Pro ucelené výsledky testů byly v prvních třech experimentech ponechány všechny kombinace pravidel, včetně dvou stejných pravidel[2].

Vytvořená matice (viz např. tabulka 8.1) byla převedena do tabulky v MS Excel a makrem byly vytvořeny konfigurační soubory pro GCCAS. Vzhledem k tomu, že vygenerovaných simulací bylo pouze 64, bylo možno je ručně spustit a klasifikovat. Ruční klasifikací se rozumí provedení předem zvoleného počtu kroků a zařazení do příslušné třídy na základě vizuálního posouzení.

Bylo odhadnuto několik parametrů, u kterých se předpokládá, že mohou ovlivnit výslednou klasifikaci globálně řízeného automatu. Vždy byly zvoleny dvě rozdílné hodnoty sledovaného parametru, byly vygenerovány konfigurační soubory a provedena klasifikace. Následně byly výsledky porovnány a zhodnocen vliv sledovaného parametru. Následující kapitoly a tabulky popisují provedené experimenty.

V prvním sloupci tabulek je uvedena třída a jednotlivá pravidla, která byla použita jako první v pořadí v simulaci. První řádek obsahuje třídy a pravidla, která byla jako druhá v simulaci. Jednotlivé buňky tabulek potom udávají číslo testu a za lomítkem třídu, do které byl automat s příslušnou kombinací pravidel klasifikován.

8.2.1 Počet kroků simulace

Sledovaným parametrem v tomto případě je celkový počet provedených výpočtů (kroků) automatu. Pro první pokus byl parametr `run` nastaven na 1 000 kroků. Výsledek shrnuje tabulka 8.1. Druhý pokus byl proveden pro 30 000 kroků, výsledek je zobrazen v tabulce 8.2. Oba experimenty vycházely ze stejné výchozí konfigurace, 50 buněk s hodnotou 0, 1 buňka s hodnotou 1 a opět 50 buněk s hodnotou 0, celkový počet buněk automatu byl 101.

Porovnáním tabulek 8.1 a 8.2 byly zjištěny dva rozdíly, v konfiguraci číslo 38 a 51, kdy při vyšším počtu kroků došlo k ustálení původně chaotického chování do stabilních struktur. K tomuto ustálení došlo po cca 1 500 krocích.

Analýza vlivu počtu kroků na vývoj chování simulace byla zařazena jako první test, protože bylo nutno určit vhodný počet kroků pro další prováděné experimenty. S počtem kroků simulace roste časová a paměťová náročnost výpočtu a také, protože změna byla zaznamenána pouze u dvou případů, byl počet kroků zvolen na 3 000. K poslední změně chování došlo v rámci tohoto testu kolem kroku 1 500, 3 000 je dvojnásobkem tohoto hraničního počtu a bude pro následující testy považován za dostačující, jak vzhledem k analýze chování, tak také vzhledem k náročnosti výpočtu.

[2]Referenční klasifikace uvedená v příloze předpokládá náhodnou počáteční konfiguraci a neomezený počet buněk automatu. V provedených testech jsou uvažovány konkrétní okrajové podmínky a jiná počáteční konfigurace a proto mohou být automaty obsahující kombinaci dvou stejných pravidel zařazeny do jiné třídy než uvedená pravidla.

Tabulka 8.1: Klasifikace automatů z experimentu 8.2.1 pro celkový počet kroků simulace 1 000.

		1		2		3		4	
		040	**128**	**228**	**109**	**022**	**045**	**054**	**110**
1	**040**	0/tř. 1	1/tř. 1	2/tř. 1	3/tř. 2	4/tř. 1	5/tř. 2	6/tř. 1	7/tř. 1
	128	8/tř. 1	9/tř. 1	10/tř. 1	11/tř. 2	12/tř. 1	13/tř. 2	14/tř. 1	15/tř. 1
2	**228**	16/tř. 1	17/tř. 1	18/tř. 2	19/tř. 2	20/tř. 3	21/tř. 3	22/tř. 4	23/tř. 3
	109	24/tř. 2	25/tř. 2	26/tř. 3	27/tř. 3	28/tř. 3	29/tř. 3	30/tř. 2	31/tř. 3
3	**022**	32/tř. 1	33/tř. 1	34/tř. 3	35/tř. 3	36/tř. 3	37/tř. 3	38/tř. 3	39/tř. 3
	045	40/tř. 2	41/tř. 2	42/tř. 3	43/tř. 3	44/tř. 3	45/tř. 3	46/tř. 3	47/tř. 3
4	**054**	48/tř. 1	49/tř. 1	50/tř. 4	51/tř. 3	52/tř. 2	53/tř. 3	54/tř. 2	55/tř. 3
	110	56/tř. 1	57/tř. 1	58/tř. 3	59/tř. 3	60/tř. 3	61/tř. 3	62/tř. 3	63/tř. 3

Tabulka 8.2: Klasifikace automatů z experimentu 8.2.1 pro celkový počet kroků simulace 30 000.

		1		2		3		4	
		040	**128**	**228**	**109**	**022**	**045**	**054**	**110**
1	**040**	0/tř. 1	1/tř. 1	2/tř. 1	3/tř. 2	4/tř. 1	5/tř. 2	6/tř. 1	7/tř. 1
	128	8/tř. 1	9/tř. 1	10/tř. 1	11/tř. 2	12/tř. 1	13/tř. 2	14/tř. 1	15/tř. 1
2	**228**	16/tř. 1	17/tř. 1	18/tř. 2	19/tř. 2	20/tř. 3	21/tř. 3	22/tř. 4	23/tř. 3
	109	24/tř. 2	25/tř. 2	26/tř. 3	27/tř. 3	28/tř. 3	29/tř. 3	30/tř. 2	31/tř. 3
3	**022**	32/tř. 1	33/tř. 1	34/tř. 3	35/tř. 3	36/tř. 3	37/tř. 3	38/tř. 2	39/tř. 3
	045	40/tř. 2	41/tř. 2	42/tř. 3	43/tř. 3	44/tř. 3	45/tř. 3	46/tř. 3	47/tř. 3
4	**054**	48/tř. 1	49/tř. 1	50/tř. 4	51/tř. 2	52/tř. 2	53/tř. 3	54/tř. 2	55/tř. 3
	110	56/tř. 1	57/tř. 1	58/tř. 3	59/tř. 3	60/tř. 3	61/tř. 3	62/tř. 3	63/tř. 3

8.2.2 Počet buněk automatu

Další experiment byl zaměřen na zkoumání vlivu počtu buněk automatu na klasifikaci vývoje. První pokus byl proveden na automatu s celkovým počtem buněk 21, počáteční konfigurace byla nastavena na 10 buněk s hodnotou 0, 1 buňka s hodnotou 1 a opět 10 buněk s hodnotou 0, okrajové podmínky byly jako u předchozího experimentu periodické. Počet kroků byl na základě předchozích zjištění nastaven na 3000 a opakování jednotlivých pravidel bylo ponecháno pro obě pravidla na 10. Druhý pokus byl proveden na počáteční konfiguraci 100 buněk s hodnotou 0, 1 buňka s hodnotou 1 a 100 buněk s hodnotou 0, celkem tedy 201 buněk. Třetí pokus na automatu s 601 buňkami, počáteční konfigurace

stejná jako v předchozích případech, jedna buňka uprostřed s hodnotou 1. Výsledky jsou shrnuty v tabulkách 8.3, 8.4 a 8.5

Tabulka 8.3: Klasifikace automatů z experimentu 8.2.2 pro 21 buněk automatu.

		1		2		3		4	
		040	**128**	**228**	**109**	**022**	**045**	**054**	**110**
1	**040**	0/tř. 1	1/tř. 1	2/tř. 1	3/tř. 2	4/tř. 1	5/tř. 2	6/tř. 1	7/tř. 1
	128	8/tř. 1	9/tř. 1	10/tř. 1	11/tř. 2	12/tř. 1	13/tř. 2	14/tř. 1	15/tř. 1
2	**228**	16/tř. 1	17/tř. 1	18/tř. 2	19/tř. 2	20/tř. 2	21/tř. 2	22/tř. 2	23/tř. 2
	109	24/tř. 2	25/tř. 2	26/tř. 3	27/tř. 2	28/tř. 3	29/tř. 2	30/tř. 2	31/tř. 3
3	**022**	32/tř. 1	33/tř. 1	34/tř. 4	35/tř. 2	36/tř. 2	37/tř. 3	38/tř. 1	39/tř. 3
	045	40/tř. 2	41/tř. 2	42/tř. 2	43/tř. 2	44/tř. 3	45/tř. 3	46/tř. 4	47/tř. 3
4	**054**	48/tř. 1	49/tř. 1	50/tř. 4	51/tř. 2	52/tř. 2	53/tř. 4	54/tř. 2	55/tř. 3
	110	56/tř. 1	57/tř. 1	58/tř. 3	59/tř. 3	60/tř. 3	61/tř. 3	62/tř. 3	63/tř. 4

Tabulka 8.4: Klasifikace automatů z experimentu 8.2.2 pro 201 buněk automatu.

		1		2		3		4	
		040	**128**	**228**	**109**	**022**	**045**	**054**	**110**
1	**040**	0/tř. 1	1/tř. 1	2/tř. 1	3/tř. 2	4/tř. 1	5/tř. 2	6/tř. 1	7/tř. 1
	128	8/tř. 1	9/tř. 1	10/tř. 1	11/tř. 2	12/tř. 1	13/tř. 2	14/tř. 1	15/tř. 1
2	**228**	16/tř. 1	17/tř. 1	18/tř. 2	19/tř. 3	20/tř. 3	21/tř. 3	22/tř. 4	23/tř. 3
	109	24/tř. 2	25/tř. 2	26/tř. 3	27/tř. 2	28/tř. 3	29/tř. 3	30/tř. 3	31/tř. 3
3	**022**	32/tř. 1	33/tř. 1	34/tř. 4	35/tř. 3	36/tř. 3	37/tř. 3	38/tř. 3	39/tř. 3
	045	40/tř. 2	41/tř. 2	42/tř. 3	43/tř. 3	44/tř. 3	45/tř. 3	46/tř. 3	47/tř. 3
4	**054**	48/tř. 1	49/tř. 1	50/tř. 4	51/tř. 3	52/tř. 3	53/tř. 3	54/tř. 2	55/tř. 3
	110	56/tř. 1	57/tř. 1	58/tř. 3	59/tř. 3	60/tř. 3	61/tř. 3	62/tř. 3	63/tř. 4

Tabulka 8.5: Klasifikace automatů z experimentu 8.2.2 pro 601 buněk automatu.

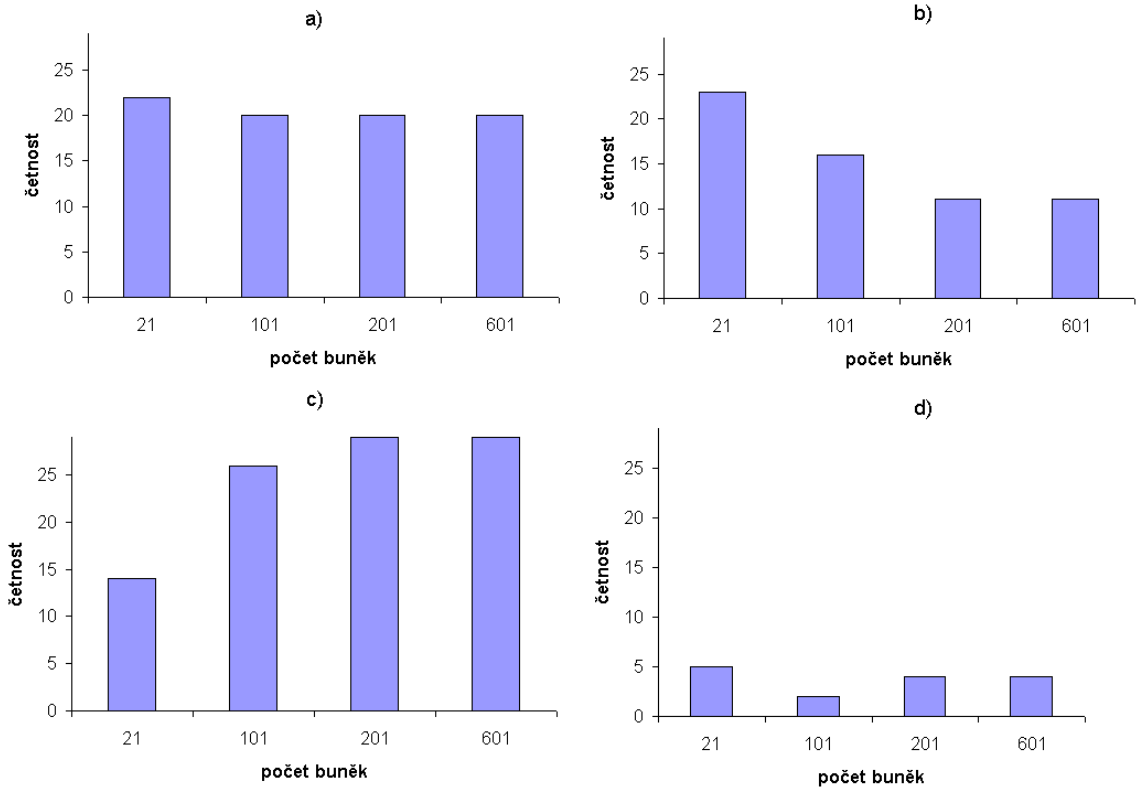
		1		2		3		4	
		040	**128**	**228**	**109**	**022**	**045**	**054**	**110**
1	**040**	0/tř. 1	1/tř. 1	2/tř. 1	3/tř. 2	4/tř. 1	5/tř. 2	6/tř. 1	7/tř. 1
	128	8/tř. 1	9/tř. 1	10/tř. 1	11/tř. 2	12/tř. 1	13/tř. 2	14/tř. 1	15/tř. 1
2	**228**	16/tř. 1	17/tř. 1	18/tř. 2	19/tř. 3	20/tř. 3	21/tř. 3	22/tř. 4	23/tř. 3
	109	24/tř. 2	25/tř. 2	26/tř. 3	27/tř. 2	28/tř. 2	29/tř. 3	30/tř. 3	31/tř. 3
3	**022**	32/tř. 1	33/tř. 1	34/tř. 3	35/tř. 3	36/tř. 3	37/tř. 3	38/tř. 3	39/tř. 3
	045	40/tř. 2	41/tř. 2	42/tř. 3	43/tř. 3	44/tř. 3	45/tř. 4	46/tř. 3	47/tř. 3
4	**054**	48/tř. 1	49/tř. 1	50/tř. 4	51/tř. 3	52/tř. 3	53/tř. 3	54/tř. 2	55/tř. 3
	110	56/tř. 1	57/tř. 1	58/tř. 3	59/tř. 3	60/tř. 3	61/tř. 3	62/tř. 3	63/tř. 4

Tabulka 8.6: Výsledek experimentu 8.2.2, počet pravidel ve Wolframových třídách v závislosti na počtu buněk automatu.

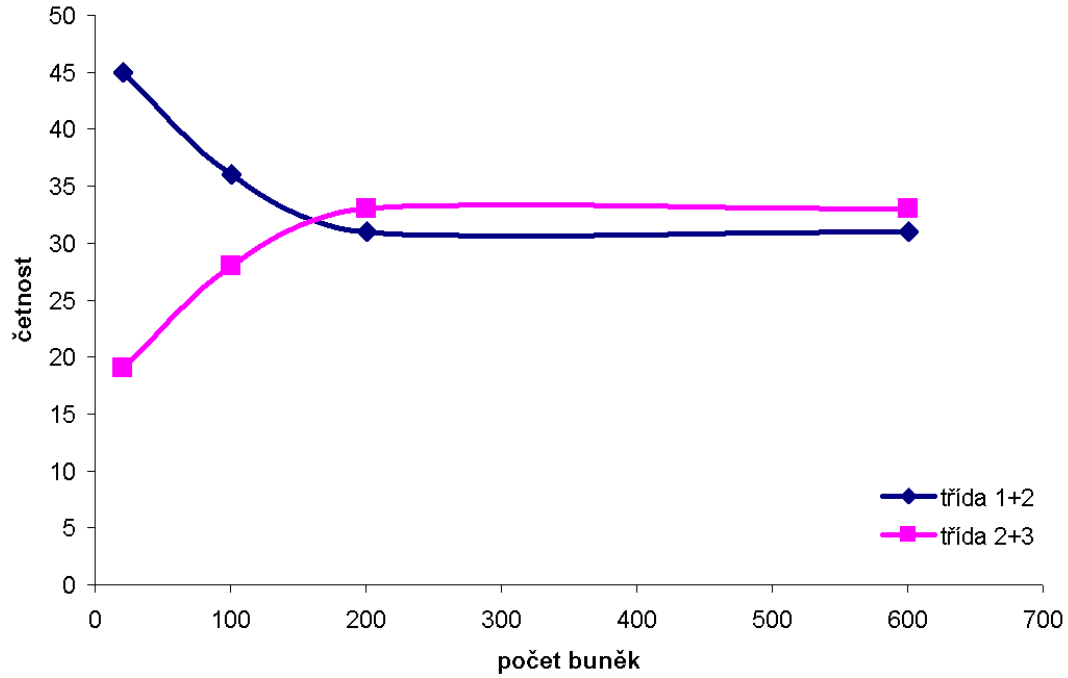
	Počet automatů			
Wolframova třída	21 buněk	101 buněk	201 buněk	601 buněk
1	22	20	20	20
2	23	16	11	11
3	14	26	29	29
4	5	2	4	4

Do srovnání provedeného experimentu byly zahrnuty také výsledky z tabulky 8.2, tím byly získány výsledky pro 21, 101, 201 a 601 buněk automatu. V tabulce 8.6 jsou shrnuty počty pravidel v jednotlivých Wolframových třídách. Tato data jsou zobrazena na souhrnném obrázku 8.2. Z obrázku je patrné, že počet automatů v první třídě je na počtu buněk automatu nezávislý.

Automaty, které jsou klasifikovány do první a druhé Wolframovy třídy, mají podobnou dynamiku vývoje, všechny setrvávají ve stejné hodnotě, nebo se opakuje krátká sekvence hodnot. Naopak automaty ze třetí a čtvrté třídy jsou charakteristické složitým až chaotickým chováním. Graf 8.3 zobrazuje počet pravidel ve sloučených třídách v závislosti na počtu buněk automatu. Z grafu je patrné, že malý počet buněk automatu vede k nízké dynamice, nárůst počtu buněk zvýší dynamiku. Omezujícím faktorem je počet automatů v první a čtvrté třídě, který je na počtu buněk nezávislý.



Obrázek 8.2: Výsledek experimentu 8.2.2, četnost pravidel ve Wolframových třídách v závislosti na počtu buněk automatu, a) 1. třída, b) 2. třída, c) 3. třída, d) 4. třída.



Obrázek 8.3: Výsledek experimentu 8.2.2, četnost pravidel ve sloučených třídách 1+2 a 2+3 v závislosti na počtu buněk automatu.

8.2.3 Typ okrajových podmínek

Další ze série experimentů byl zaměřen na vliv typu okrajových podmínek na klasifikaci automatu. Byly provedeny dva testy, oba pro fixní okrajové podmínky, první pro hodnotu 0 a druhý pro 1. Počet výpočtů byl nastaven stejně jako v předchozím případě na 3 000 a na základě předchozího experimentu byl počet buněk nastaven na 201, což zajišťuje dostatek prostoru pro rozvinutí dynamiky automatu. Jako třetí test byla použita data z tabulky 8.4, která byla získána testy s automaty s periodickými okrajovými podmínkami. Tímto byly analyzovány všechny tři implementované typy okrajových podmínek.

Tabulka 8.7: Klasifikace automatů z experimentu 8.2.3 pro fixní okrajové podmínky s hodnotou 0.

		1		2		3		4	
		040	**128**	**228**	**109**	**022**	**045**	**054**	**110**
1	**040**	0/tř. 1	1/tř. 1	2/tř. 1	3/tř. 2	4/tř. 1	5/tř. 2	6/tř. 1	7/tř. 1
	128	8/tř. 1	9/tř. 1	10/tř. 1	11/tř. 2	12/tř. 1	13/tř. 2	14/tř. 1	15/tř. 1
2	**228**	16/tř. 1	17/tř. 1	18/tř. 2	19/tř. 3	20/tř. 3	21/tř. 3	22/tř. 2	23/tř. 3
	109	24/tř. 2	25/tř. 2	26/tř. 3	27/tř. 3	28/tř. 3	29/tř. 2	30/tř. 3	31/tř. 3
3	**022**	32/tř. 1	33/tř. 1	34/tř. 3	35/tř. 3	36/tř. 3	37/tř. 3	38/tř. 2	39/tř. 3
	045	40/tř. 2	41/tř. 2	42/tř. 2	43/tř. 2	44/tř. 3	45/tř. 4	46/tř. 3	47/tř. 3
4	**054**	48/tř. 1	49/tř. 1	50/tř. 2	51/tř. 3	52/tř. 2	53/tř. 3	54/tř. 2	55/tř. 2
	110	56/tř. 1	57/tř. 1	58/tř. 3	59/tř. 3	60/tř. 3	61/tř. 3	62/tř. 3	63/tř. 2

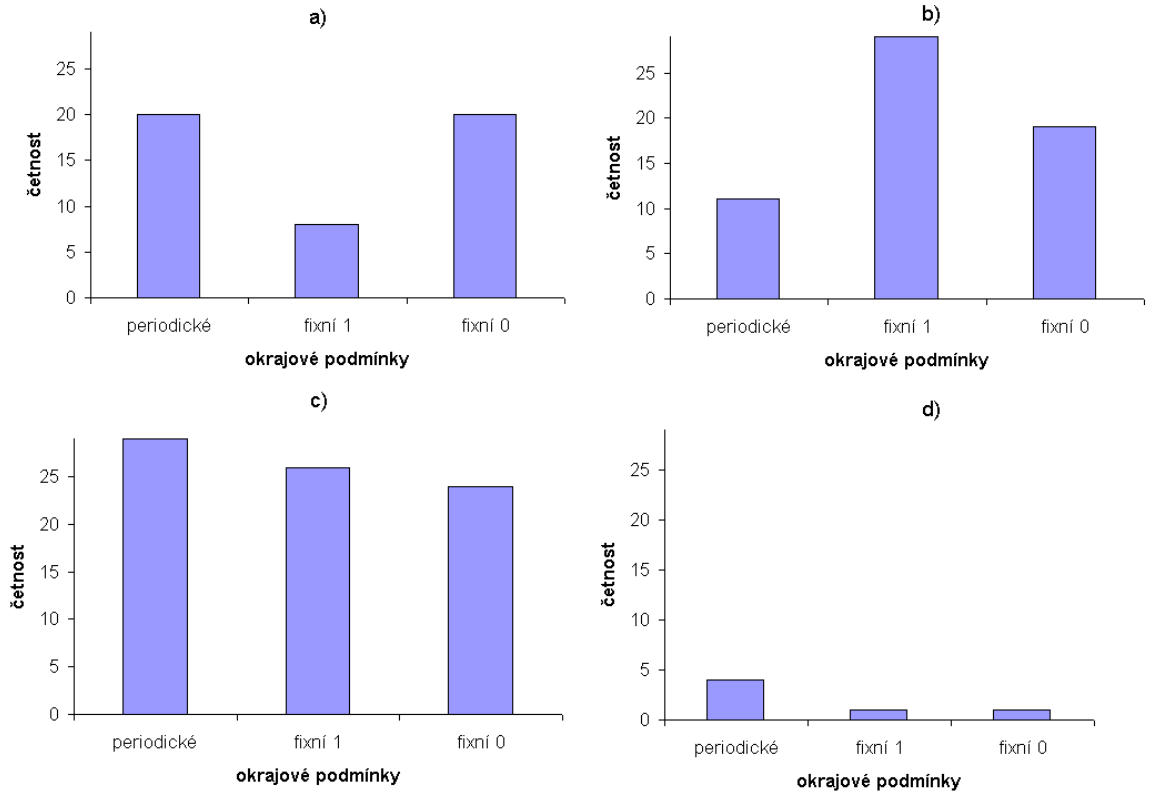
Tabulka 8.8: Klasifikace automatů z experimentu 8.2.3 pro fixní okrajové podmínky s hodnotou 1.

		1		2		3		4	
		040	**128**	**228**	**109**	**022**	**045**	**054**	**110**
1	**040**	0/tř. 1	1/tř. 1	2/tř. 1	3/tř. 2	4/tř. 2	5/tř. 2	6/tř. 2	7/tř. 2
	128	8/tř. 1	9/tř. 1	10/tř. 1	11/tř. 2	12/tř. 2	13/tř. 2	14/tř. 2	15/tř. 2
2	**228**	16/tř. 1	17/tř. 1	18/tř. 2	19/tř. 3	20/tř. 3	21/tř. 2	22/tř. 2	23/tř. 3
	109	24/tř. 2	25/tř. 2	26/tř. 3	27/tř. 3	28/tř. 3	29/tř. 2	30/tř. 3	31/tř. 3
3	**022**	32/tř. 2	33/tř. 2	34/tř. 3	35/tř. 3	36/tř. 3	37/tř. 3	38/tř. 3	39/tř. 3
	045	40/tř. 2	41/tř. 2	42/tř. 2	43/tř. 2	44/tř. 3	45/tř. 4	46/tř. 3	47/tř. 3
4	**054**	48/tř. 2	49/tř. 2	50/tř. 2	51/tř. 3	52/tř. 3	53/tř. 3	54/tř. 2	55/tř. 3
	110	56/tř. 2	57/tř. 2	58/tř. 3	59/tř. 3	60/tř. 3	61/tř. 3	62/tř. 3	63/tř. 2

Tabulka 8.9: Výsledek experimentu 8.2.3, četnost pravidel ve Wolframových třídách v závislosti na typu okrajových podmínek.

Wolframova třída	**Počet automatů**		
	periodické	fixní 1	fixní 0
1	20	8	20
2	11	29	19
3	29	26	24
4	4	1	1

Výsledky testů s pevnými okrajovými podmínkami jsou v tabulkách 8.7 a 8.8. Tabulka 8.9 zobrazuje četnost automatů v jednotlivých třídách v závislosti na zvolených okrajových podmínkách, výsledky z této tabulky jsou poté shrnuty v grafu 8.4. Z uvedených grafů je patrné, že největšího útlumu dynamiky celulárního automatu se dosáhne fixními okrajovými podmínkami s hodnotou 0. Okrajové podmínky s pevnou hodnotu 1 způsobí posunutí částí automatů z první třídy do druhé, což je způsobeno právě virtuálními buňkami s hodnotou 1, které mohou inicializovat další vývoj buněk. Periodické podmínky nejlépe podporují složitý až chaotický vývoj.



Obrázek 8.4: Výsledek experimentu 8.2.3, četnost pravidel v jednotlivých třídách v závislosti na typu okrajových podmínek, a) 1. třída, b) 2. třída, c) 3. třída, d) 4. třída.

8.2.4 Počet opakování pravidla

Cílem čtvrtého experimentu bylo zjistit, jaký vliv na vývoj automatu má střídání pravidel, resp. počet opakování pravidla. Referenční data pro tento experiment byla vybrána z tabulky 8.5, tedy automaty s dostatečným počtem buněk i kroků výpočtu.

Společné nastavení pro následující testy je: periodické okrajové podmínky, počáteční konfigurace 201 buněk (100x0,1x1,100x0), 3 000 kroků výpočtu. Pro první test byl nastaven počet opakování prvního pravidla na 10 a druhého pravidla na 1. Automaty byly klasifikovány a výsledky porovnány proti referenční tabulce a byl sledován počet změn, tedy počet případů, kdy vložení jednoho výpočtu jiného pravidla způsobilo změnu ve vývoji automatu. Výsledek prvního testu je zobrazen v tabulce 8.10 [3]. Druhý test obsahoval 10 opakování prvního pravidla a 5 opakování druhého pravidla. Výsledky jsou v tabulce 8.11. Za třetí test můžeme považovat referenční tabulku, která je vytvořena na základě automatů s 10 opakováním prvního i druhého pravidla. A čtvrtý test byl proveden pro 10 opakování prvního pravidla a 30 opakování druhého pravidla. Výsledky posledního testu jsou v tabulce 8.12.

Tabulka 8.10: Klasifikace automatů z experimentu 8.2.4 pro první pravidlo 10x a druhé pravidlo 1x.

		1		2		3		4	
		040	**128**	**228**	**109**	**022**	**045**	**054**	**110**
1	**040**			2/tř. 1	3/tř. 2	4/tř. 1	5/tř. 2	6/tř. 1	7/tř. 1
	128			10/tř. 1	11/tř. 2	12/tř. 1	13/tř. 2	14/tř. 1	15/tř. 1
2	**228**	16/tř. 1	17/tř. 1			20/tř. 4	21/tř. 4	22/tř. 4	23/tř. 2
	109	24/tř. 3	25/tř. 3			28/tř. 3	29/tř. 3	30/tř. 3	31/tř. 3
3	**022**	32/tř. 1	33/tř. 1	34/tř. 3	35/tř. 3			38/tř. 3	39/tř. 3
	045	40/tř. 3	41/tř. 4	42/tř. 3	43/tř. 3			46/tř. 3	47/tř. 3
4	**054**	48/tř. 1	49/tř. 1	50/tř. 4	51/tř. 2	52/tř. 3	53/tř. 3		
	110	56/tř. 3	57/tř. 4	58/tř. 3	59/tř. 3	60/tř. 3	61/tř. 3		

Tabulka 8.13 zobrazuje výsledné počty jinak klasifikovaných automatů v závislosti na počtu opakování druhého pravidla. Rozdíly jsou vztaženy vůči referenčnímu testu, pro který je uvedena nulová změna. Graf 8.5 zobrazuje tato data graficky a je patrné, že největší změnu udělá pravidlo i s jedním opakováním, větší počet opakování již nemá na klasifikaci významný vliv.

[3] Kombinace stejných pravidel byly v tomto testu vynechány, protože pro porovnání změn již nemají význam.

Tabulka 8.11: Klasifikace automatů z experimentu 8.2.4 pro první pravidlo 10x, druhé pravidlo 5x.

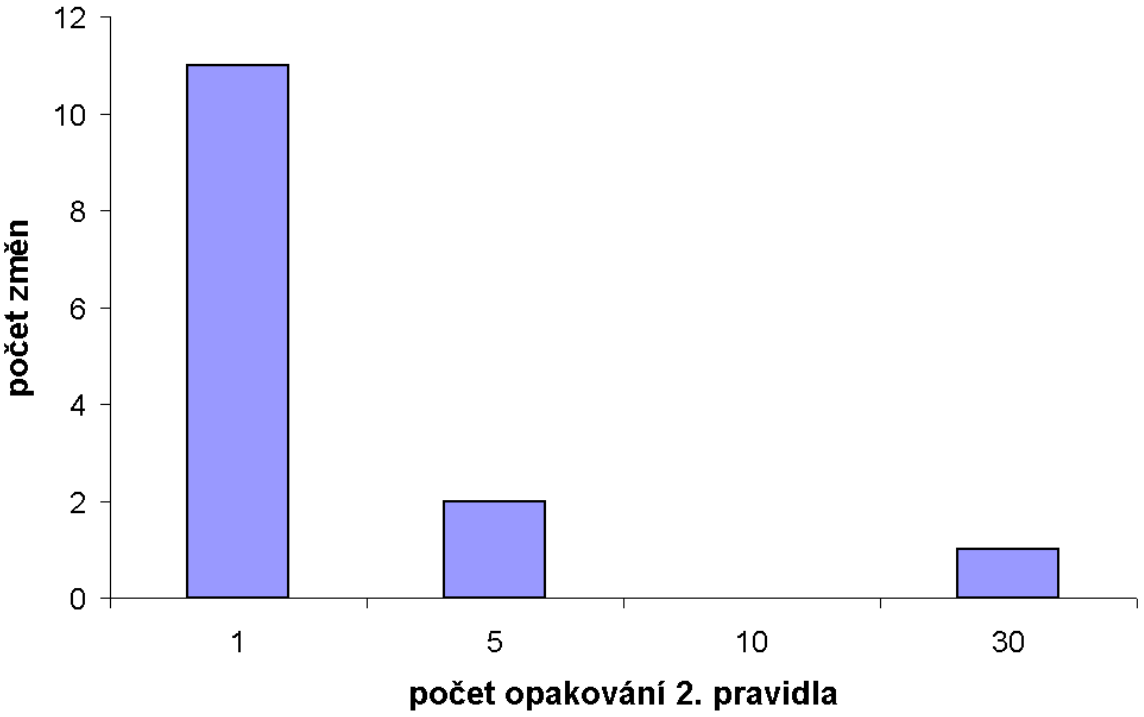
		1		2		3		4	
		040	**128**	**228**	**109**	**022**	**045**	**054**	**110**
1	**040**			2/tř. 1	3/tř. 2	4/tř. 1	5/tř. 2	6/tř. 1	7/tř. 1
	128			10/tř. 1	11/tř. 2	12/tř. 1	13/tř. 2	14/tř. 1	15/tř. 1
2	**228**	16/tř. 1	17/tř. 1			20/tř. 3	21/tř. 3	22/tř. 4	23/tř. 2
	109	24/tř. 2	25/tř. 2			28/tř. 3	29/tř. 3	30/tř. 3	31/tř. 3
3	**022**	32/tř. 1	33/tř. 1	34/tř. 3	35/tř. 3			38/tř. 3	39/tř. 3
	045	40/tř. 2	41/tř. 2	42/tř. 3	43/tř. 3			46/tř. 3	47/tř. 3
4	**054**	48/tř. 3	49/tř. 1	50/tř. 3	51/tř. 3	52/tř. 3	53/tř. 3		
	110	56/tř. 1	57/tř. 1	58/tř. 3	59/tř. 3	60/tř. 3	61/tř. 3		

Tabulka 8.12: Klasifikace automatů z experimentu 8.2.4 pro první pravidlo 10x a druhé pravidlo 30x.

		1		2		3		4	
		040	**128**	**228**	**109**	**022**	**045**	**054**	**110**
1	**040**			2/tř. 1	3/tř. 2	4/tř. 1	5/tř. 2	6/tř. 1	7/tř. 1
	128			10/tř. 1	11/tř. 2	12/tř. 1	13/tř. 2	14/tř. 1	15/tř. 1
2	**228**	16/tř. 1	17/tř. 1			20/tř. 3	21/tř. 3	22/tř. 4	23/tř. 3
	109	24/tř. 2	25/tř. 2			28/tř. 3	29/tř. 3	30/tř. 3	31/tř. 3
3	**022**	32/tř. 1	33/tř. 1	34/tř. 3	35/tř. 3			38/tř. 2	39/tř. 3
	045	40/tř. 2	41/tř. 2	42/tř. 3	43/tř. 3			46/tř. 3	47/tř. 3
4	**054**	48/tř. 1	49/tř. 1	50/tř. 4	51/tř. 3	52/tř. 3	53/tř. 3		
	110	56/tř. 1	57/tř. 1	58/tř. 3	59/tř. 3	60/tř. 3	61/tř. 3		

Tabulka 8.13: Výsledek experimentu 8.2.4, počet jinak klasifikovaných automatů v závislosti na počtu opakování druhého pravidla.

Počet opakování	1	5	10	30
Počet změn	11	2	0	1



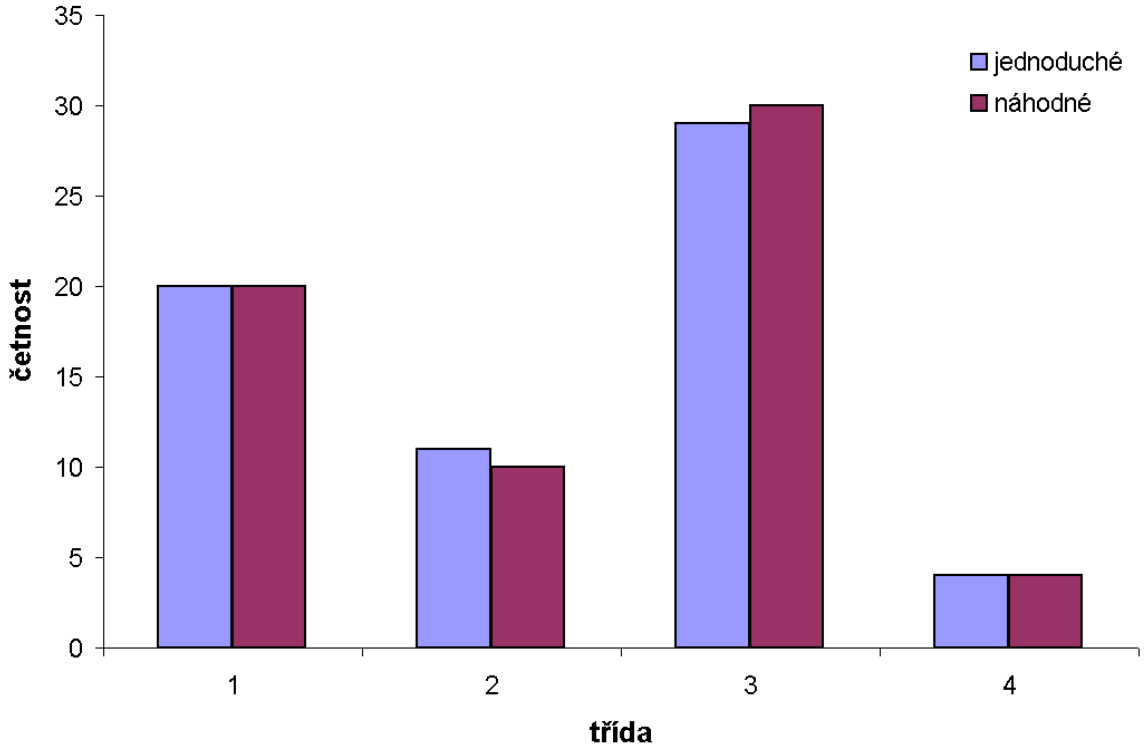
Obrázek 8.5: Výsledek experimentu 8.2.4, počet jinak klasifikovaných automatů vůči referenční tabulce.

8.2.5 Počáteční konfigurace

Cílem posledního experimentu bylo ověřit, jestli a jak počáteční konfigurace ovlivňuje klasifikaci automatu. Test byl proveden s automatem o 201 buňkách, náhodnou počáteční konfigurací, periodickými okrajovými podmínkami a 3 000 výpočty. Výsledek testu je zobrazen v tabulce 8.14. Pro srovnání byly použity výsledky z tabulky 8.4, tedy stejné podmínky, ale jiná počáteční konfigurace. Obě pravidla se opakovala shodně 10x.

Graf 8.6 zobrazuje četnost pravidel v jednotlivých třídách pro náhodnou a jednoduchou[4] počáteční konfigurací. Náhodná počáteční konfigurace způsobila pouze dva jinak klasifikované automaty, nemá tedy příliš velký vliv na klasifikaci.

[4] Jedna buňka s hodnotou 1 uprostřed, tzn. například 100x0 1x1 100x0.



Obrázek 8.6: Výsledek experimentu 8.2.4, závislost klasifikace automatů na počáteční konfiguraci.

Tabulka 8.14: Klasifikace automatů z experimentu 8.2.5 pro náhodnou počáteční konfiguraci.

		1		2		3		4	
		040	**128**	**228**	**109**	**022**	**045**	**054**	**110**
1	**040**	0/tř. 1	1/tř. 1	2/tř. 1	3/tř. 2	4/tř. 1	5/tř. 2	6/tř. 1	7/tř. 1
	128	8/tř. 1	9/tř. 1	10/tř. 1	11/tř. 2	12/tř. 1	13/tř. 2	14/tř. 1	15/tř. 1
2	**228**	16/tř. 1	17/tř. 1	18/tř. 2	19/tř. 3	20/tř. 3	21/tř. 3	22/tř. 4	23/tř. 3
	109	24/tř. 2	25/tř. 2	26/tř. 3	27/tř. 2	28/tř. 3	29/tř. 3	30/tř. 3	31/tř. 3
3	**022**	32/tř. 1	33/tř. 1	34/tř. 3	35/tř. 3	36/tř. 3	37/tř. 3	38/tř. 3	39/tř. 3
	045	40/tř. 2	41/tř. 2	42/tř. 3	43/tř. 3	44/tř. 3	45/tř. 3	46/tř. 3	47/tř. 3
4	**054**	48/tř. 1	49/tř. 1	50/tř. 4	51/tř. 3	52/tř. 3	53/tř. 3	54/tř. 4	55/tř. 3
	110	56/tř. 1	57/tř. 1	58/tř. 3	59/tř. 3	60/tř. 3	61/tř. 3	62/tř. 3	63/tř. 4

8.3 Souhrn experimentů a ověření předpokladů

8.3.1 Souhrn

Tato kapitola je souhrnem experimentů z kapitoly 8.2. Ke zkoumaným parametrům globálně řízených celulárních automatů je na základě provedených experimentů přidána informace o jejich vlivu na klasifikaci vývoje.

Poznatky z provedených testů jsou souhrnně zobrazeny v grafech 8.7 - 8.10, které procentuelně vyjadřují četnost automatů v jednotlivých třídách v závislosti na zkoumaném parametru. Grafy byly vytvořeny na základě tabulky 8.15 resp. 8.16, která obsahuje data ze všech provedených experimentů.

Návrh charakterizace globálně řízených celulárních automatů je založen na uvedených grafech, resp. na míře vlivu jednotlivých parametrů na klasifikaci vývoje automatů, která z grafů plyne. Předpokladem návrhu je zachování rozložení četnosti automatů do tříd v závislosti na parametrech automatu. Závislost na zvolené kombinaci pravidel je omezena na vlastnost obživnutí, která ze své podstaty snižuje počet automatů zařazených do první třídy.

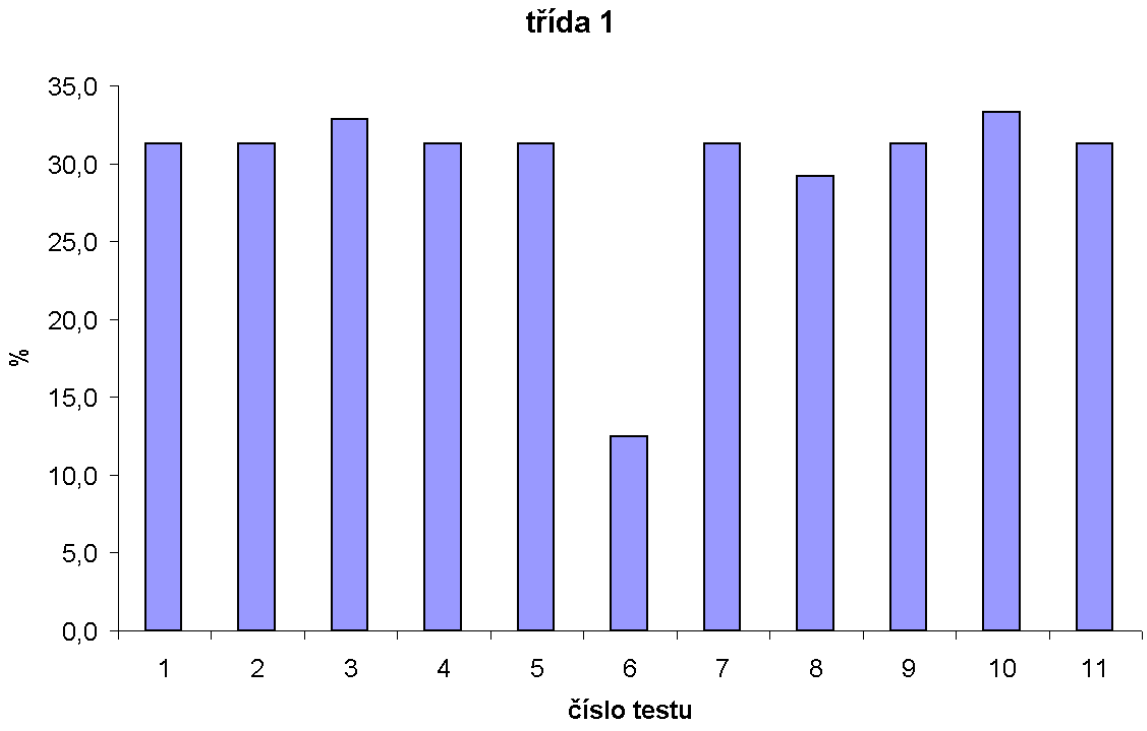
Četnost automatů v 1. třídě nejvíce ovlivňují fixní okrajové podmínky s hodnotou 1, které sníží četnost automatů proti ostatním parametrům na polovinu. Fixní okrajové podmínky s hodnotou 1 se projevily opačným vlivem ve 2. třídě, kde způsobily naopak největší četnost. Dalším faktorem ovlivňujícím klasifikaci do 2. třídy je počet buněk celulárního automatu. Nedostatečný počet buněk snižuje možnost rozvoje dynamiky, a proto bylo 37,5% automatů klasifikováno do 2. třídy. Třetí významný faktor pro 2. třídu jsou fixní okrajové podmínky s hodnotou 0. Četnost automatů ve 3. třídě kolísá mezi 47,9% až 37,5% s jedním minimem s 21,9% při nízkém (21) počtu buněk automatu. Největší počet složitých nebo přesunujících se struktur, tj. 12,5% automatů vzniklo při pouze jednom opakování druhého pravidla.

Tabulka 8.15: Souhrn provedených experimentů.

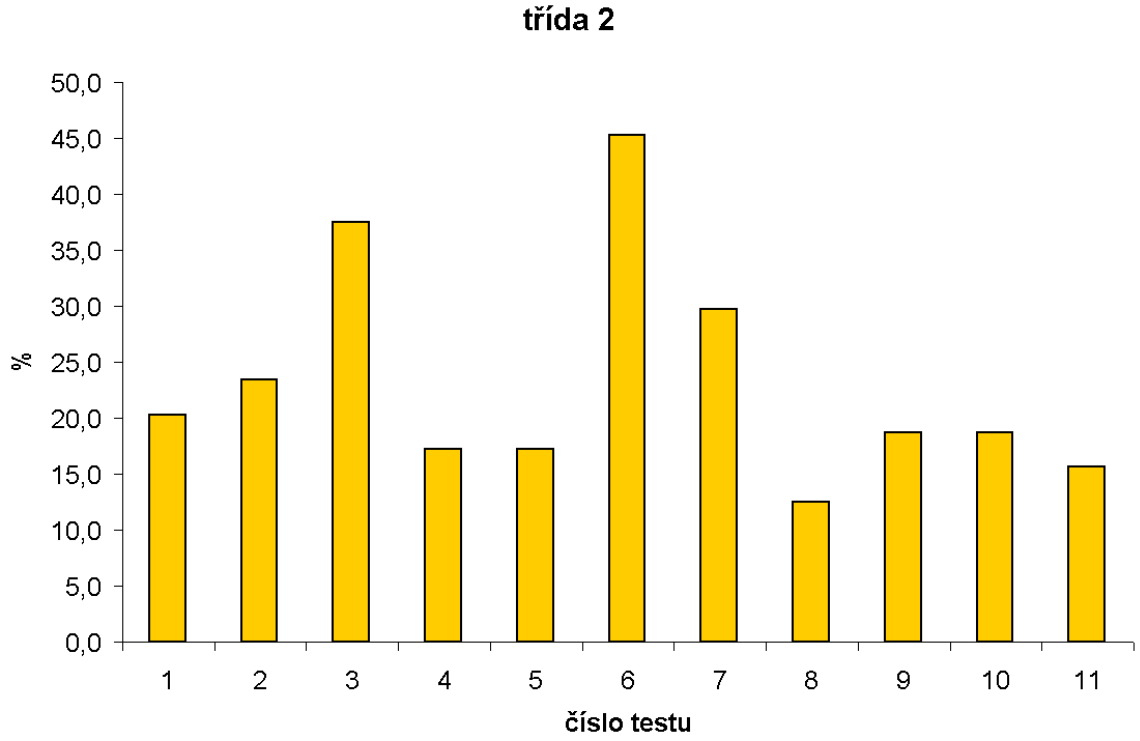
parametr/číslo testu	1	2	3	4	5	6
okrajové podm.	per.	per.	per.	per.	per.	fix1
počet buněk	101	101	21	201	601	201
počet kroků	1000	30 000	3 000	3 000	3 000	3 000
počáteční konf.	jed.	jed.	jed.	jed.	jed.	jed.
počet 1. pravidla	10	10	10	10	10	10
počet 2. pravidla	10	10	10	10	10	10
třída	% automatů					
1	31.3	31.3	32.8	31.3	31.3	12.5
2	20.3	23.4	37.5	17.2	17.2	45.3
3	45.3	42.2	21.9	45.3	45.3	40.6
4	3.1	3.1	7.8	6.3	6.3	1.6

Tabulka 8.16: Pokračování tabulky 8.15.

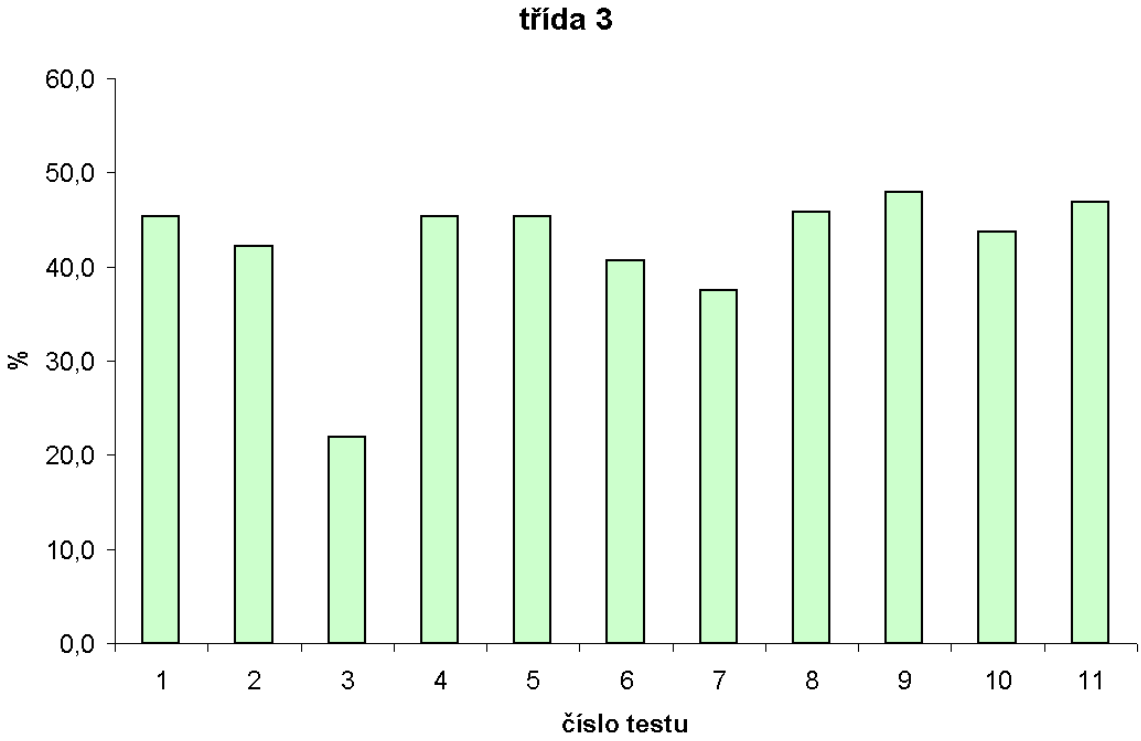
parametr/číslo testu	7	8	9	10	11
okrajové podm.	fix0	per.	per.	per.	per.
počet buněk	201	201	201	201	201
počet kroků	3 000	3 000	3 000	3 000	3 000
počáteční konf.	jed.	jed,	jed.	jed.	náh.
počet 1. pravidla	10	10	10	10	10
počet 2. pravidla	10	1	5	30	10
třída	% automatů				
1	31.3	29.2	31.3	33.3	31.3
2	29.7	12.5	18.8	18.8	15.6
3	37.5	45.8	47.9	43.8	46.9
4	1.6	12.5	2.1	4.2	6.3



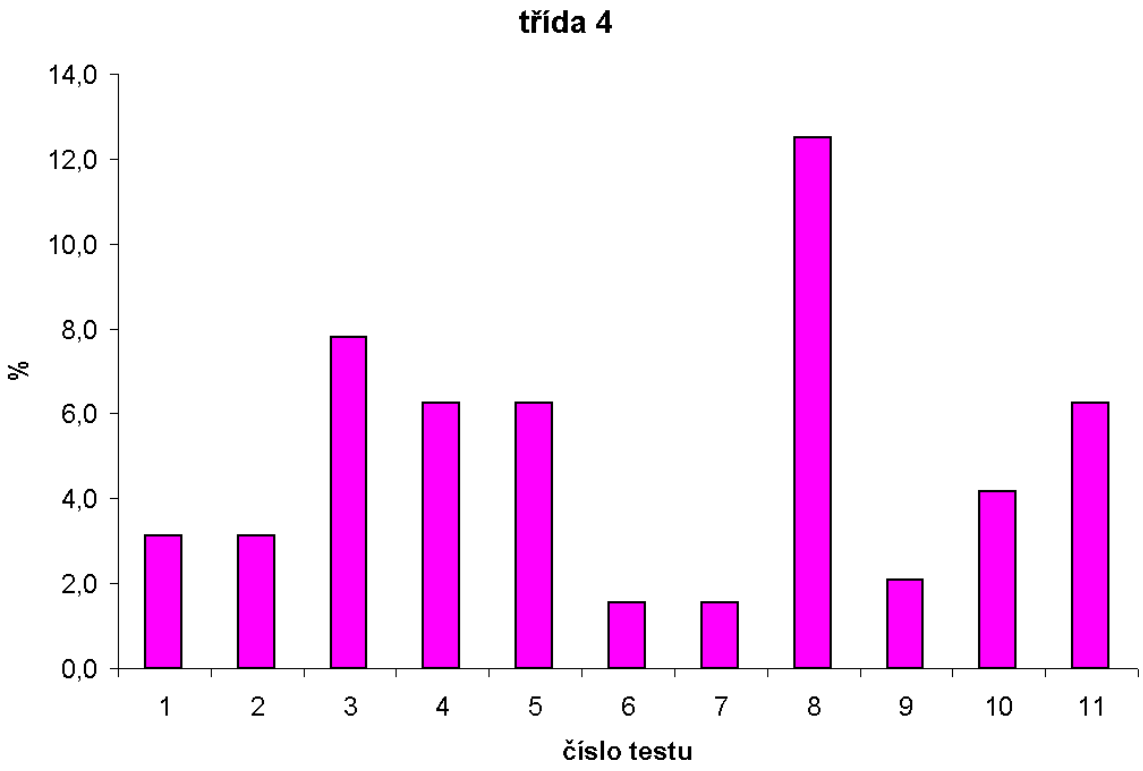
Obrázek 8.7: Souhrn procentuální četnosti automatů v závislosti na zvolených parametrech v 1. třídě.



Obrázek 8.8: Souhrn procentuální četnosti automatů v závislosti na zvolených parametrech ve 2. třídě.



Obrázek 8.9: Souhrn procentuální četnosti automatů v závislosti na zvolených parametrech ve 3. třídě.



Obrázek 8.10: Souhrn procentuální četnosti automatů v závislosti na zvolených parametrech ve 4. třídě.

8.3.2 Ověření předpokladu: Modelový případ

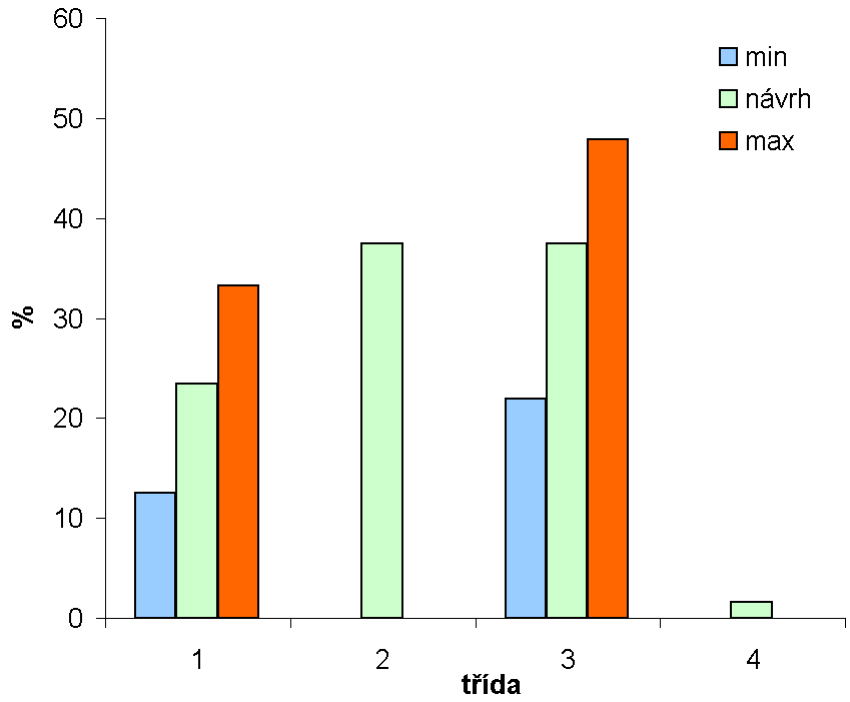
Pro vyzkoušení platnosti předpokladů plynoucích z provedených experimentů byla provedena klasifikace jiné osmice pravidel. Zadání modelového případu: K dispozici je sada globálně řízených celulárních automatů s pevně definovanými pravidly a nastavitelnými parametry. Navrhněte tyto parametry tak, aby nejvíce automatů bylo ve třetí třídě a nejméně v první.

Podle uvedených požadavků bylo navrženo: 201 buněk automatu, náhodná počáteční konfigurace, fixní okrajové podmínky s hodnotou 1, počet opakování prvního pravidla 10x, druhého 5x a 3 000 výpočtů. Takto navržené automaty byly klasifikovány a výsledky jsou uvedeny v tabulce 8.17.

Výsledek klasifikace modelového případu je zobrazen také v grafu 8.11. Pro první a třetí třídu, na kterou byly kladeny specifické požadavky, byly přidány minimální a maximální hodnoty četnosti zjištěné v rámci všech experimentálních testů. Oba typy krajních hodnot byly zjištěny při částečně odlišných podmínkách proti modelovému případu, kde byly parametry nastaveny jako kombinace dvou požadavků, což se projevilo na výsledcích. Četnost automatů v první a třetí třídě proto nedosahuje krajních hodnot. Navržený výsledek je tedy kompromisem dvou požadavků.

Tabulka 8.17: Výsledná klasifikace modelového případu.

		1		2		3		4	
		253	**239**	**21**	**123**	**60**	**129**	**147**	**137**
1	**253**	0/tř. 1	1/tř. 1	2/tř. 2	3/tř. 2	4/tř. 2	5/tř. 1	6/tř. 1	7/tř. 1
	239	8/tř. 1	9/tř. 1	10/tř. 2	11/tř. 2	12/tř. 2	13/tř. 2	14/tř. 1	15/tř. 1
2	**21**	16/tř. 2	17/tř. 2	18/tř. 2	19/tř. 2	20/tř. 2	21/tř. 2	22/tř. 2	23/tř. 2
	123	24/tř. 2	25/tř. 2	26/tř. 2	27/tř. 2	28/tř. 3	29/tř. 3	30/tř. 3	31/tř. 3
3	**60**	32/tř. 2	33/tř. 2	34/tř. 3	35/tř. 2	36/tř. 3	37/tř. 3	38/tř. 3	39/tř. 3
	129	40/tř. 1	41/tř. 1	42/tř. 3	43/tř. 3	44/tř. 3	45/tř. 3	46/tř. 3	47/tř. 3
4	**147**	48/tř. 1	49/tř. 1	50/tř. 3	51/tř. 3	52/tř. 3	53/tř. 3	54/tř. 4	55/tř. 3
	137	56/tř. 1	57/tř. 1	58/tř. 2	59/tř. 3	60/tř. 3	61/tř. 3	62/tř. 3	63/tř. 2



Obrázek 8.11: Porovnání minimálních a maximálních hodnot z experimentů proti modelovému případu.

Kapitola 9

Diskuse

V této kapitole je položeno několik otevřených otázek souvisejících s problematikou globálního řízení celulárních automatů a jejich charakterizací. Otázky byly tvořeny v průběhu zpracovávání této práce. Uvedené odpovědi jsou pouze zamyšlením nad danou otázkou a jejich úplné zodpovězení je otevřeno dalším autorům v této problematice.

Má globální řízení celulárních automatů vliv na kvalitu generovaných pseudonáhodných čísel?
Vliv globálního řízení celulárních automatů na jejich využití jako generátorů pseudonáhodných čísel nebyl v rámci této diplomové práce analyzován.

Mohou globálně řízené automaty generovat vzory, které není možno klasickým celulárním automatem vytvořit?
Ano, základním příkladem je schopnost pokračování vývoje generovaného vzoru v případě, kdy všechny buňky automatu mají hodnotu 0, a dojde k přepnutí na pravidlo s vlastností obživnutí.

Je možno provedené experimenty, jejich výsledky a obecně vlastnosti globálně řízených automatů popsat matematickými vztahy?
Literatura uvádí techniky pro matematický popis základní třídy binárních 1D celulárních automatů, jejich implementace ale není součástí této práce. Je možno tedy předpokládat, že také globálně řízené celulární automaty by bylo možno popsat matematickými vztahy.

Je možno vizuální klasifikaci nahradit softwarovým řešením?
Minimálně částečně ano, automaty v první a druhé třídě. Testováním jestli všechny buňky nabývají hodnoty 0 nebo 1 po určitém konečném počtu výpočtů pro třídu 1 a testováním jestli se vzor posledních x výpočtů opakuje pro druhou třídu. Automatické rozpoznání automatů ve třetí a čtvrté třídě souvisí s předchozí otázkou na matematický popis. V případě úspěšné implementace matematické charakteristiky je možno předpokládat automatickou charakterizaci do zbylých dvou tříd.

Je referenční tabulka pravidel vhodně zvolena?
Zařazení základních 256 pravidel do Wolframových tříd byl věnován značný čas při zpracovávání této práce. Vzhledem k rozporům v literatuře, byl zvolen zdroj Wolfram Alpha, pod jehož vytvořením je podepsán Stephen Wolfram[1] a který je z prostudovaných zdrojů nejmladší. Byly předpokládány informace obsahující nejvíce poznatků.

[1] autor knihy A new kind of science[21]

Kapitola 10

Závěr

Cílem této diplomové práce bylo seznámit se s problematikou celulárních automatů a možností globálního řízení. Na základě těchto znalostí navrhnout a implementovat vhodný simulační nástroj a charakterizovat 1D celulární automaty. Dále navrhnout způsob charakterizace globálně řízených celulárních automatů a pomocí vytvořeného nástroje experimentálně tyto automaty charakterizovat.

Vytvořený simulátor byl implementován podle matematické definice globálně řízených 1D celulárních automatů. Pro charakterizace takovýchto automatů byla navržena série experimentů, které měly za úkol zjistit, do jaké míry ovlivňují parametry globálně řízeného celulárního automatu jeho dynamiku, resp. do jaké Wolframovy třídy bude klasifikován. Experimenty testovaly vliv těchto parametrů automatů: počet provedených výpočtů, počet buněk automatu, typ okrajových podmínek, počet výpočtů jednoho pravidla a nastavení počátečních hodnot buněk. Výsledky jednotlivých experimentů byly průběžně zobrazovány v grafech a po provedení všech experimentů byly tyto výsledky shrnuty do přehledné tabulky a čtyř grafů, pro každou Wolframovu třídu jeden. Z těchto grafů je dobře patrné, jaký vliv mají jednotlivé parametry celulárního automatu na jeho dynamiku. Nejvýraznější změny jsou patrné v první třídě, kdy fixní okrajové podmínky s hodnotou 1 výrazně sníží četnost těchto automatů, dojde k jejich přesunu do druhé třídy. Další výrazný vliv na dynamiku má počet buněk automatu, kdy při 21 buňkách dojde k silnému útlumu dynamiky a přesunu automatů ze třetí třídy do druhé. Četnost automatů ve čtvrté třídě je obecně nízká, výrazné změny bylo dosaženo pouze jedním opakováním druhého pravidla, což vneslo do generovaného vzoru "ruchy"které se dále šířily.

Navržená charakterizace se zakládá na provedených experimentech, tedy na předpokladu, že primární vliv na charakterizaci mají parametry automatu a výběr pravidla v rámci jedné Wolframovy třídy je potlačen. Platnost tohoto předpokladu byla zjišťována v rámci modelového případu, ve kterém byly dostupné globálně řízené celulární automaty s pevně danými pravidly, a úkolem bylo na základě požadavků vhodně určit parametry. Pravidla použitá v modelovém případu byla záměrně zvolena jiná než při experimentálních testech a také požadavky byly zadány tak, aby nevyhovovaly přímo některému prováděnému experimentu. Klasifikací navržených automatů bylo zjištěno, že zjištěné předpoklady fungují, ale při kombinaci požadavků dojde ke kompromisu ve výsledcích, tzn. parametry se navzájem ovlivňují a nepodařilo se při jejich kombinaci dosáhnout krajních hodnot tak, jako při jednotlivých experimentech.

V kapitole 9 (Diskuse) bylo nastíněno několik otevřených otázek, které vyplynuly při vypracovávání této diplomové práce. Na některé z nich byly navrženy odpovědi ve smyslu zamyšlení se nad daným problémem. Otázky z této kapitoly by mohly být počátečním

bodem pro další experimenty či výzkum v této oblasti. Především by se mohlo jednat o matematickou (automatickou) klasifikaci globálně řízených celulárních automatů nebo o analýzu vlivu globálního řízení na kvalitu generovaných pseudonáhodných čísel.

Literatura

[1] Creutz M.: Xtoys. `http://thy.phy.bnl.gov/www/xtoys/xtoys.html`, cit. 2009-01-01.

[2] Dave, B.: *Generative Music and Cellular Automata. PhD Thesis.* Univ. Technology Sydney, Australia, 2006.

[3] Floreano, D.: *Bio-inspired artificial intelligence: theories,methods, and technologies.* Massachusetts Institute of Technology, 2008.

[4] Haase, C.; Guy, R.: *Filthy rich clients : developing animated and graphical effects for desktop Java applications.* Sun Microsystems, Inc., 2008.

[5] Jelínek L.: Java (24) - úvod do grafiky a GUI. `http://www.linuxsoft.cz/article.php?id_article=1184`, cit. 2010-04-18.

[6] Kari J.: Cellular Automata. `http://users.utu.fi/jkari/ca/`, cit. 2009-01-01.

[7] Korček, P.: *Generovanie pseudonáhodných čísel v FPGA, bakalárska práca.* FIT VUT v Brne, 2007.

[8] Mařík, V.: *Umělá inteligence.[Díl] 3 / 1. vyd.* Praha: Academia, 2001.

[9] Peringer, P.: SNT - Celularní automaty (doplněk), výukový materiál kurzu SNT. `https://www.fit.vutbr.cz/study/courses/SNT/public/Prednasky/SNT-cellular.pdf`, cit. 2009-12-12.

[10] Robert, E.; Marca, L.; Dave, W.; aj.: *Java Swing 2 Rev Ed.* O'Reilly Media, Inc, 2002.

[11] Sekanina, L.: *Evolvable components: from theory to hardware implementations.* Berlin: Springer-Verlag, 2004.

[12] Sekanina, L.: Development v evolučním návrhu, výukový materiál kurzu BIN. `https://www.fit.vutbr.cz/study/courses/BIN/private/`, cit. 2009-01-01.

[13] Sekanina, L.; Komenda, T.: *Global Control In Polymorphic Cellular Automata. Journal of Cellular Automata.* 2010 v tisku.

[14] Sun Microsystems: Learn About Java Technology. `http://www.java.com/en/download/faq/whatis_java.xml`, cit. 2010-04-15.

[15] Tang T.; Wishart D.: Erik Max Francis. `http://www.alcyone.com/software/cage/`, cit. 2009-01-01.

[16] Tang T.; Wishart D.: SimCell.
http://wishart.biology.ualberta.ca/SimCell/index.html, cit. 2009-01-01.

[17] Weimar J.R.: JCASim: Cellular automata simulation system.
http://www.jcasim.de/, cit. 2009-01-01.

[18] Wikipedia: Swing (Java). http://cs.wikipedia.org/wiki/Swing_Java, cit. 2010-04-15.

[19] Wikipedia: Java (programming language).
http://en.wikipedia.org/wiki/Java_programming_language, cit. 2010-04-16.

[20] Wikipedia: Standard Widget Toolkit.
http://cs.wikipedia.org/wiki/Standard_Widget_Toolkit, cit. 2010-04-19.

[21] Wolfram, S.: *A New Kind of Science*. Wolfram Media, 2002.

[22] Wolfram Research, Inc.: Wolfram Atlas: Rule properties:Example.
http://atlas.wolfram.com/01/01/views/87/TableView.html, cit. 2009-01-01.

[23] Wolfram Research, Inc.: WolframAplha computational, knowledge engine.
http://www.wolframalpha.com/, cit. 2010-04-08.

[24] Wuensche, A.; Lesser, M.: *The global dynamics of cellular automata: an atlas of basin of attraction fields of one-dimensional cellular automata*. Addison-Wesley Publishing Company, 1992.

[25] Wuensche Andy: Classifying Cellular Automata Automatically: Finding gliders, filtering, and relating space-time patterns, attractor basins, and the Z parameter.
http://www.cogs.susx.ac.uk/users/andywu/publications.html, 1999 cit. 2010-04-08.

[26] WWW stránky: Celulární automat - Takzvané kluzákové dělo.
http://cs.wikipedia.org/wiki/Soubor:Gospers_glider_gun.gif, cit. 2009-01-01.

[27] WWW stránky: jednofuché struktury v LIFE.
http://cs.wikipedia.org/wiki/Soubor:Life.png, cit. 2009-01-01.

Příloha A

První spuštění GCCAS

Tato příloha popisuje jak rychle a snadno vyzkoušet simulátor globálně řízených celulárních automatů GCCAS, který byl v rámci této diplomové práce vytvořen. Následující kroky popisují první spuštění simulátoru a provedení vzorové simulace.

- předpokládá se počítač s funkčním Java Runtime Enviroment ve verzi 6u20 (nebo vyšší, kompatibilní), grafický výstup na monitor s rozlišením 800x600 a vyšším,
- v adresáři `gccas_jar` na přiloženém CD spustit příkaz:
  `java -jar gccas.jar`,
- z menu Simulation vybrat položku Load from file vybrat z adresáře `gccas_jar` soubor `vzor.sim`,
- proběhne načtení simulace a automaticky se provede 100 kroků výpočtů.

Bližší popis uživatelské rozhraní aplikace je uveden v kapitole 6.4.1 této diplomové práce.

Příloha B

Obsah CD

Tato příloha popisuje obsah přiloženého CD. Následuje popis adresářové struktury se stručným vysvětlením obsahu.

- `dp_pdf` - výsledný text této diplomové práce ve formátu PDF,
- `dp_src` - zdrojové texty této diplomové práce,
- `gccas_jar` - výsledný soubor implementovaného simulátoru,
- `gccas_src` - zdrojové kódy simulátoru,
- `sim` - definiční soubory simulací, které byly provedeny v rámci této diplomové práce,
- `tools` - nástroj pro generování sady simulačních souborů.

Příloha C

Referenční tabulka 256 pravidel

Tato příloha obsahuje referenční tabulku základních 256 pravidel jednorozměrného celulárního automatu.

Tabulka C.1: Referenční vlastnosti pravidel pro binární 1D CA s okolím r=1

Pravidlo	binárně								Z	λ	třída	obživnutí
0	0	0	0	0	0	0	0	0	0,00	0,000	1	
1	0	0	0	0	0	0	0	1	0,25	0,125	2	ano
2	0	0	0	0	0	0	1	0	0,25	0,125	2	
3	0	0	0	0	0	0	1	1	0,50	0,250	2	ano
4	0	0	0	0	0	1	0	0	0,25	0,125	2	
5	0	0	0	0	0	1	0	1	0,50	0,250	2	ano
6	0	0	0	0	0	1	1	0	0,50	0,250	2	
7	0	0	0	0	0	1	1	1	0,75	0,375	2	ano
8	0	0	0	0	1	0	0	0	0,25	0,125	1	
9	0	0	0	0	1	0	0	1	0,50	0,250	2	ano
10	0	0	0	0	1	0	1	0	0,50	0,250	2	
11	0	0	0	0	1	0	1	1	0,75	0,375	2	ano
12	0	0	0	0	1	1	0	0	0,50	0,250	2	
13	0	0	0	0	1	1	0	1	0,75	0,375	2	ano
14	0	0	0	0	1	1	1	0	0,75	0,375	2	
15	0	0	0	0	1	1	1	1	1,00	0,500	2	ano
16	0	0	0	1	0	0	0	0	0,25	0,125	2	
17	0	0	0	1	0	0	0	1	0,50	0,250	2	ano
18	0	0	0	1	0	0	1	0	0,50	0,250	3	
19	0	0	0	1	0	0	1	1	0,63	0,375	2	ano
20	0	0	0	1	0	1	0	0	0,50	0,250	2	
21	0	0	0	1	0	1	0	1	0,75	0,375	2	ano
22	0	0	0	1	0	1	1	0	0,75	0,375	3	
23	0	0	0	1	0	1	1	1	0,50	0,500	2	ano
24	0	0	0	1	1	0	0	0	0,50	0,250	2	
25	0	0	0	1	1	0	0	1	0,75	0,375	2	ano
26	0	0	0	1	1	0	1	0	0,75	0,375	2	
27	0	0	0	1	1	0	1	1	0,75	0,500	2	ano
28	0	0	0	1	1	1	0	0	0,75	0,375	2	

Pravidlo	binárně								Z	λ	třída	obživnutí
29	0	0	0	1	1	1	0	1	0,50	0,500	2	ano
30	0	0	0	1	1	1	1	0	1,00	0,500	3	
31	0	0	0	1	1	1	1	1	0,75	0,625	2	ano
32	0	0	1	0	0	0	0	0	0,25	0,125	1	
33	0	0	1	0	0	0	0	1	0,50	0,250	2	ano
34	0	0	1	0	0	0	1	0	0,50	0,250	2	
35	0	0	1	0	0	0	1	1	0,63	0,375	2	ano
36	0	0	1	0	0	1	0	0	0,50	0,250	2	
37	0	0	1	0	0	1	0	1	0,75	0,375	2	ano
38	0	0	1	0	0	1	1	0	0,75	0,375	2	
39	0	0	1	0	0	1	1	1	0,75	0,500	2	ano
40	0	0	1	0	1	0	0	0	0,50	0,250	1	
41	0	0	1	0	1	0	0	1	0,75	0,375	4	ano
42	0	0	1	0	1	0	1	0	0,75	0,375	2	
43	0	0	1	0	1	0	1	1	0,50	0,500	2	ano
44	0	0	1	0	1	1	0	0	0,75	0,375	2	
45	0	0	1	0	1	1	0	1	1,00	0,500	3	ano
46	0	0	1	0	1	1	1	0	0,50	0,500	2	
47	0	0	1	0	1	1	1	1	0,75	0,625	2	ano
48	0	0	1	1	0	0	0	0	0,50	0,250	2	
49	0	0	1	1	0	0	0	1	0,63	0,375	2	ano
50	0	0	1	1	0	0	1	0	0,63	0,375	2	
51	0	0	1	1	0	0	1	1	1,00	0,500	2	ano
52	0	0	1	1	0	1	0	0	0,75	0,375	2	
53	0	0	1	1	0	1	0	1	0,75	0,500	2	ano
54	0	0	1	1	0	1	1	0	0,75	0,500	4	
55	0	0	1	1	0	1	1	1	0,63	0,625	2	ano
56	0	0	1	1	1	0	0	0	0,75	0,375	2	
57	0	0	1	1	1	0	0	1	0,75	0,500	2	ano
58	0	0	1	1	1	0	1	0	0,75	0,500	2	
59	0	0	1	1	1	0	1	1	0,63	0,625	2	ano
60	0	0	1	1	1	1	0	0	1,00	0,500	3	
61	0	0	1	1	1	1	0	1	0,75	0,625	2	ano
62	0	0	1	1	1	1	1	0	0,75	0,625	2	
63	0	0	1	1	1	1	1	1	0,50	0,750	2	ano
64	0	1	0	0	0	0	0	0	0,25	0,125	1	
65	0	1	0	0	0	0	0	1	0,50	0,250	2	ano
66	0	1	0	0	0	0	1	0	0,50	0,250	2	
67	0	1	0	0	0	0	1	1	0,75	0,375	2	ano
68	0	1	0	0	0	1	0	0	0,50	0,250	2	
69	0	1	0	0	0	1	0	1	0,75	0,375	2	ano
70	0	1	0	0	0	1	1	0	0,75	0,375	2	
71	0	1	0	0	0	1	1	1	0,50	0,500	2	ano
72	0	1	0	0	1	0	0	0	0,50	0,250	2	
73	0	1	0	0	1	0	0	1	0,75	0,375	2	ano
74	0	1	0	0	1	0	1	0	0,75	0,375	2	
75	0	1	0	0	1	0	1	1	1,00	0,500	3	ano

Pravidlo	binárně								Z	λ	třída	obživnutí
76	0	1	0	0	1	1	0	0	0,63	0,375	2	
77	0	1	0	0	1	1	0	1	0,50	0,500	2	ano
78	0	1	0	0	1	1	1	0	0,75	0,500	2	
79	0	1	0	0	1	1	1	1	0,75	0,625	2	ano
80	0	1	0	1	0	0	0	0	0,50	0,250	2	
81	0	1	0	1	0	0	0	1	0,75	0,375	2	ano
82	0	1	0	1	0	0	1	0	0,75	0,375	2	
83	0	1	0	1	0	0	1	1	0,75	0,500	2	ano
84	0	1	0	1	0	1	0	0	0,75	0,375	2	
85	0	1	0	1	0	1	0	1	1,00	0,500	2	ano
86	0	1	0	1	0	1	1	0	1,00	0,500	3	
87	0	1	0	1	0	1	1	1	0,75	0,625	2	ano
88	0	1	0	1	1	0	0	0	0,75	0,375	2	
89	0	1	0	1	1	0	0	1	1,00	0,500	3	ano
90	0	1	0	1	1	0	1	0	1,00	0,500	3	
91	0	1	0	1	1	0	1	1	0,75	0,625	2	ano
92	0	1	0	1	1	1	0	0	0,75	0,500	2	
93	0	1	0	1	1	1	0	1	0,75	0,625	2	ano
94	0	1	0	1	1	1	1	0	0,75	0,625	2	
95	0	1	0	1	1	1	1	1	0,50	0,750	2	ano
96	0	1	1	0	0	0	0	0	0,50	0,250	1	
97	0	1	1	0	0	0	0	1	0,75	0,375	2	ano
98	0	1	1	0	0	0	1	0	0,75	0,375	2	
99	0	1	1	0	0	0	1	1	0,75	0,500	2	ano
100	0	1	1	0	0	1	0	0	0,75	0,375	2	
101	0	1	1	0	0	1	0	1	1,00	0,500	3	ano
102	0	1	1	0	0	1	1	0	1,00	0,500	3	
103	0	1	1	0	0	1	1	1	0,75	0,625	2	ano
104	0	1	1	0	1	0	0	0	0,75	0,375	2	
105	0	1	1	0	1	0	0	1	1,00	0,500	3	ano
106	0	1	1	0	1	0	1	0	1,00	0,500	4	
107	0	1	1	0	1	0	1	1	0,75	0,625	2	ano
108	0	1	1	0	1	1	0	0	0,75	0,500	2	
109	0	1	1	0	1	1	0	1	0,75	0,625	2	ano
110	0	1	1	0	1	1	1	0	0,75	0,625	4	
111	0	1	1	0	1	1	1	1	0,50	0,750	2	ano
112	0	1	1	1	0	0	0	0	0,75	0,375	2	
113	0	1	1	1	0	0	0	1	0,50	0,500	2	ano
114	0	1	1	1	0	0	1	0	0,75	0,500	2	
115	0	1	1	1	0	0	1	1	0,63	0,625	2	ano
116	0	1	1	1	0	1	0	0	0,50	0,500	2	
117	0	1	1	1	0	1	0	1	0,75	0,625	2	ano
118	0	1	1	1	0	1	1	0	0,75	0,625	2	
119	0	1	1	1	0	1	1	1	0,50	0,750	2	ano
120	0	1	1	1	1	0	0	0	1,00	0,500	4	
121	0	1	1	1	1	0	0	1	0,75	0,625	4	ano
122	0	1	1	1	1	0	1	0	0,75	0,625	3	

Pravidlo	binárně								Z	λ	třída	obživnutí
123	0	1	1	1	1	0	1	1	0,50	0,750	2	ano
124	0	1	1	1	1	1	0	0	0,75	0,625	4	
125	0	1	1	1	1	1	0	1	0,50	0,750	2	ano
126	0	1	1	1	1	1	1	0	0,50	0,750	3	
127	0	1	1	1	1	1	1	1	0,25	0,875	2	ano
128	1	0	0	0	0	0	0	0	0,25	0,125	1	
129	1	0	0	0	0	0	0	1	0,50	0,250	3	ano
130	1	0	0	0	0	0	1	0	0,50	0,250	2	
131	1	0	0	0	0	0	1	1	0,75	0,375	2	ano
132	1	0	0	0	0	1	0	0	0,50	0,250	2	
133	1	0	0	0	0	1	0	1	0,75	0,375	2	ano
134	1	0	0	0	0	1	1	0	0,75	0,375	2	
135	1	0	0	0	0	1	1	1	1,00	0,500	3	ano
136	1	0	0	0	1	0	0	0	0,50	0,250	1	
137	1	0	0	0	1	0	0	1	0,75	0,375	4	ano
138	1	0	0	0	1	0	1	0	0,75	0,375	2	
139	1	0	0	0	1	0	1	1	0,50	0,500	2	ano
140	1	0	0	0	1	1	0	0	0,63	0,375	2	
141	1	0	0	0	1	1	0	1	0,75	0,500	2	ano
142	1	0	0	0	1	1	1	0	0,50	0,500	2	
143	1	0	0	0	1	1	1	1	0,75	0,625	2	ano
144	1	0	0	1	0	0	0	0	0,50	0,250	2	
145	1	0	0	1	0	0	0	1	0,75	0,375	2	ano
146	1	0	0	1	0	0	1	0	0,75	0,375	3	
147	1	0	0	1	0	0	1	1	0,75	0,500	4	ano
148	1	0	0	1	0	1	0	0	0,75	0,375	2	
149	1	0	0	1	0	1	0	1	1,00	0,500	3	ano
150	1	0	0	1	0	1	1	0	1,00	0,500	3	
151	1	0	0	1	0	1	1	1	0,75	0,625	3	ano
152	1	0	0	1	1	0	0	0	0,75	0,375	2	
153	1	0	0	1	1	0	0	1	1,00	0,500	3	ano
154	1	0	0	1	1	0	1	0	1,00	0,500	2	
155	1	0	0	1	1	0	1	1	0,75	0,625	2	ano
156	1	0	0	1	1	1	0	0	0,75	0,500	2	
157	1	0	0	1	1	1	0	1	0,75	0,625	2	ano
158	1	0	0	1	1	1	1	0	0,75	0,625	2	
159	1	0	0	1	1	1	1	1	0,50	0,750	2	ano
160	1	0	1	0	0	0	0	0	0,50	0,250	1	
161	1	0	1	0	0	0	0	1	0,75	0,375	3	ano
162	1	0	1	0	0	0	1	0	0,75	0,375	2	
163	1	0	1	0	0	0	1	1	0,75	0,500	2	ano
164	1	0	1	0	0	1	0	0	0,75	0,375	2	
165	1	0	1	0	0	1	0	1	1,00	0,500	3	ano
166	1	0	1	0	0	1	1	0	1,00	0,500	2	
167	1	0	1	0	0	1	1	1	0,75	0,625	2	ano
168	1	0	1	0	1	0	0	0	0,75	0,375	1	
169	1	0	1	0	1	0	0	1	1,00	0,500	4	ano

Pravidlo	binárně								Z	λ	třída	obživnutí
170	1	0	1	0	1	0	1	0	1,00	0,500	2	
171	1	0	1	0	1	0	1	1	0,75	0,625	2	ano
172	1	0	1	0	1	1	0	0	0,75	0,500	2	
173	1	0	1	0	1	1	0	1	0,75	0,625	2	ano
174	1	0	1	0	1	1	1	0	0,75	0,625	2	
175	1	0	1	0	1	1	1	1	0,50	0,750	2	ano
176	1	0	1	1	0	0	0	0	0,75	0,375	2	
177	1	0	1	1	0	0	0	1	0,75	0,500	2	ano
178	1	0	1	1	0	0	1	0	0,50	0,500	2	
179	1	0	1	1	0	0	1	1	0,63	0,625	2	ano
180	1	0	1	1	0	1	0	0	1,00	0,500	2	
181	1	0	1	1	0	1	0	1	0,75	0,625	2	ano
182	1	0	1	1	0	1	1	0	0,75	0,625	3	
183	1	0	1	1	0	1	1	1	0,50	0,750	3	ano
184	1	0	1	1	1	0	0	0	0,50	0,500	2	
185	1	0	1	1	1	0	0	1	0,75	0,625	2	ano
186	1	0	1	1	1	0	1	0	0,75	0,625	2	
187	1	0	1	1	1	0	1	1	0,50	0,750	2	ano
188	1	0	1	1	1	1	0	0	0,75	0,625	2	
189	1	0	1	1	1	1	0	1	0,50	0,750	2	ano
190	1	0	1	1	1	1	1	0	0,50	0,750	2	
191	1	0	1	1	1	1	1	1	0,25	0,875	2	ano
192	1	1	0	0	0	0	0	0	0,50	0,250	1	
193	1	1	0	0	0	0	0	1	0,75	0,375	4	ano
194	1	1	0	0	0	0	1	0	0,75	0,375	2	
195	1	1	0	0	0	0	1	1	1,00	0,500	3	ano
196	1	1	0	0	0	1	0	0	0,63	0,375	2	
197	1	1	0	0	0	1	0	1	0,75	0,500	2	ano
198	1	1	0	0	0	1	1	0	0,75	0,500	2	
199	1	1	0	0	0	1	1	1	0,75	0,625	2	ano
200	1	1	0	0	1	0	0	0	0,63	0,375	2	
201	1	1	0	0	1	0	0	1	0,75	0,500	2	ano
202	1	1	0	0	1	0	1	0	0,75	0,500	2	
203	1	1	0	0	1	0	1	1	0,75	0,625	2	ano
204	1	1	0	0	1	1	0	0	1,00	0,500	2	
205	1	1	0	0	1	1	0	1	0,63	0,625	2	ano
206	1	1	0	0	1	1	1	0	0,63	0,625	2	
207	1	1	0	0	1	1	1	1	0,50	0,750	2	ano
208	1	1	0	1	0	0	0	0	0,75	0,375	2	
209	1	1	0	1	0	0	0	1	0,50	0,500	2	ano
210	1	1	0	1	0	0	1	0	1,00	0,500	2	
211	1	1	0	1	0	0	1	1	0,75	0,625	2	ano
212	1	1	0	1	0	1	0	0	0,50	0,500	2	
213	1	1	0	1	0	1	0	1	0,75	0,625	2	ano
214	1	1	0	1	0	1	1	0	0,75	0,625	2	
215	1	1	0	1	0	1	1	1	0,50	0,750	2	ano
216	1	1	0	1	1	0	0	0	0,75	0,500	2	

Pravidlo	binárně								Z	λ	třída	obživnutí
217	1	1	0	1	1	0	0	1	0,75	0,625	2	ano
218	1	1	0	1	1	0	1	0	0,75	0,625	2	
219	1	1	0	1	1	0	1	1	0,50	0,750	2	ano
220	1	1	0	1	1	1	0	0	0,63	0,625	2	
221	1	1	0	1	1	1	0	1	0,50	0,750	2	ano
222	1	1	0	1	1	1	1	0	0,50	0,750	2	
223	1	1	0	1	1	1	1	1	0,25	0,875	2	ano
224	1	1	1	0	0	0	0	0	0,75	0,375	1	
225	1	1	1	0	0	0	0	1	1,00	0,500	4	ano
226	1	1	1	0	0	0	1	0	0,50	0,500	2	
227	1	1	1	0	0	0	1	1	0,75	0,625	2	ano
228	1	1	1	0	0	1	0	0	0,75	0,500	2	
229	1	1	1	0	0	1	0	1	0,75	0,625	2	ano
230	1	1	1	0	0	1	1	0	0,75	0,625	2	
231	1	1	1	0	0	1	1	1	0,50	0,750	2	ano
232	1	1	1	0	1	0	0	0	0,50	0,500	2	
233	1	1	1	0	1	0	0	1	0,75	0,625	2	ano
234	1	1	1	0	1	0	1	0	0,75	0,625	1	
235	1	1	1	0	1	0	1	1	0,50	0,750	1	ano
236	1	1	1	0	1	1	0	0	0,63	0,625	2	
237	1	1	1	0	1	1	0	1	0,50	0,750	2	ano
238	1	1	1	0	1	1	1	0	0,50	0,750	1	
239	1	1	1	0	1	1	1	1	0,25	0,875	1	ano
240	1	1	1	1	0	0	0	0	1,00	0,500	2	
241	1	1	1	1	0	0	0	1	0,75	0,625	2	ano
242	1	1	1	1	0	0	1	0	0,75	0,625	2	
243	1	1	1	1	0	0	1	1	0,50	0,750	2	ano
244	1	1	1	1	0	1	0	0	0,75	0,625	2	
245	1	1	1	1	0	1	0	1	0,50	0,750	2	ano
246	1	1	1	1	0	1	1	0	0,50	0,750	2	
247	1	1	1	1	0	1	1	1	0,25	0,875	2	ano
248	1	1	1	1	1	0	0	0	0,75	0,625	1	
249	1	1	1	1	1	0	0	1	0,50	0,750	1	ano
250	1	1	1	1	1	0	1	0	0,50	0,750	1	
251	1	1	1	1	1	0	1	1	0,25	0,875	1	ano
252	1	1	1	1	1	1	0	0	0,50	0,750	1	
253	1	1	1	1	1	1	0	1	0,25	0,875	1	ano
254	1	1	1	1	1	1	1	0	0,25	0,875	1	
255	1	1	1	1	1	1	1	1	0,00	1,000	1	ano