Katedra Informatiky
Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského, Bratislava

Game of Life na šesťuholníkovej mriežke

(Bakalárska práca)

Ivana Selečéniová

Vedúci: RNDr. Michal Forišek

Bratislava, 2009

Game of Life na šesťuholníkovej mriežke

Bakalárska práca

Ivana Selečéniová

UNIVERZITA KOMENSKÉHO, BRATISLAVA
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
KATEDRA INFORMATIKY

9.2.1 Informatika

Vedúci: RNDr. Michal Forišek

Bratislava 2009

Čestne prehlasujem, že som túto bakalársku prácu vypracoval(a) samostatne s použitím citovaných zdrojov.

.....

Poďakovanie

Chcela by som poďakovať vedúcemu práce RNDr. Michalovi Foriškovi za peknú tému, cenné rady a usmerňovanie počas písania práce.

Ďalej by som chcela poďakovať môjmu priateľovi za jeho pomoc a možnosť využitia jeho počítača, vďaka ktorému bolo testovanie oveľa rýchlejšie.

Moja vďaka patrí tiež rodine za ich podporu.

Abstrakt

V práci sme sa zaoberali bunkovými automatmi na šesťuholníkovej mriežke. Hlavným cieľom práce bola analýza vhodných pravidiel pre šesťuholníkový variant hry. Zamerali sme sa na hľadanie špeciálnych typov vzorov, ktoré existujú v originálnej hre. V práci sme vylúčili nevhodné pravidlá na základe Conwayových podmienok. Pre zvyšné pravidlá sme hľadali „vesmírne lode“ (vzory, ktoré sa pohybujú po mriežke) pomocou vlastného programu. Na záver práce sme uviedli nájdené vzory pre rôzne pravidlá.

Kľúčové slová: bunkové automaty, šesťuholníková mriežka, Game of Life, vesmírne lode.

Predhovor

Model bunkových automatov je v súčastnosti intenzívne skúmaný, o čom svedčí aj množstvo každoročne vydaných článkov pokrývajúcich túto tému.

Jednou z prvých aplikácií bunkových automatov, ktorá spopularizovala tento model, je Conwayova hra Game of Life. Táto hra, ktorá používa bunkový automat na štvorcovej mriežke, ponúka jednoduchý model živých organizmov.

Napriek mnohým snahám hra Game of Life nebola uspokojivo rozšírená na iné typy mriežok, aj keď z teoretického hľadiska je zaujímavé nájsť závislosť medzi typom mriežky a pravidlami pre požadované vlastnosti hry.

V práci sme sa snažili vyhovieť pôvodným trom Conwayovým podmienkam. Vytvorili sme program simulujúci šesťuholníkové bunkové automaty a testovali sme rôzne pravidlá na rôznych počiatočných konfiguráciách.

Obsah

Zoznam obrázkov

Kapitola 1

Úvod

V prírode sa často vyskytujú organizmy, ktoré sú zložené z viacerých jednoduchých častí. Tento princíp používa aj výpočtový model, ktorým sa v práci zaoberáme – bunkový automat. Napriek jednoduchosti jednotlivých buniek automatu je tento model rovnako výpočtovo silný ako turingov stroj. Tento model sa uplatnil v mnohých oblastiach od biológie až po sociálne vedy.

Jedným z výsledkov v teórii bunkových automatov bol Conwayov objav bunkového automatu pre hru Game of Life, ktorá spĺňa tri jednoduché podmienky modelujúce živé organizmy. Conwayovi sa podarilo vymyslieť automat, ktorý používa typ pravidiel závislý len od počtu živých resp. mŕtvych susedov, pričom susedom bunky je každá bunka, ktorá je s danou bunkou spojená hranou alebo vrcholom.

V práci sa zaoberáme tým, ako treba Conwayov automat zmeniť tak, aby fungoval (spĺňal Conwayove podmienky) na inej ako štvorcovej mriežke. Konkrétne skúmame automaty na šesťuholníkovej mriežke. Zvolili sme rovnako intuitívny výber susedov ako pôvodne Conway – susedmi sú bunky spojené hranou, keďže žiadne dve bunky nemôžu byť spojené iba vrcholom.

V Kapitole 2 popíšeme model bunkového automatu, neformálne aj formálne. Uvedieme stručnú históriu a prehľad aplikácií bunkových automatov. Na záver kapitoly rozoberáme možnosti voľby parametrov v definícii.

V Kapitole 3 popisujeme hru Game of Life. Uvádzame Conwayove kritériá, ktorými sa budeme riadiť pri hľadaní ekvivalentu ku Game of Life na šesťuholníkovej mriežke. Podávame prehľad rôznych typov vzorov, ako aj konkrétne známe príklady počiatočných konfigurácií pre každý typ vzoru. V kapitole tiež formálne definujeme bunkový automat zodpovedajúci Conwayovej Game of Life.

V Kapitole 4 je popísaný program simulujúci šesťuholníkové bunkové automaty s najjednoduchším typom okolia. Tento program používame v práci na testovanie pravidiel.

V Kapitole 5 aplikujeme Conwayove kritériá na pravidlá pre šesťuholníkové bunkové automaty. Pomocou prvých dvoch kritérii obmedzíme počet možných pravidiel pre šesťuholníkovú Game of Life tak, aby sme mohli zvyšné pravidlá experimentálne skúmať pomocou testovaní.

V Kapitole 6 testujeme pravidlá vzhľadom na tretie Conwayove kritérium, zameriavame sa hlavne na hľadanie „vesmírnych lodí“. Na záver tejto kapitoly uvádzame výsledky testovaní a možných kandidátov na pravidlá pre šesťuholníkový variant hry Game of Life.

Kapitola 2

Prehľad bunkových automatov

V tejto kapitole najskôr popíšeme neformálne a potom formálne štruktúru bunkových automatov. Formálna definícia je prevzatá z [Wik05] a upravená podľa našich potrieb. Z histórie uvádzame dôvody vzniku bunkových automatov a tiež autorov, ktorí majú na ich vzniku najväčšiu zásluhu. Neskôr tiež spomíname široké využitie bunkových automatov v biológii, fyzike, chémii ale aj v sociálnych odboroch a hrách. Najznámejšia hra sa volá Game of Life a čitateľ nájde jej popis v Kapitole 3. Na záver kapitoly budeme skúmať parametre z definície CA.

2.1 Definícia bunkového automatu

Najskôr neformálne popíšeme bunkové automaty. Jednorozmerný bunkový automat (*cellular automaton*, ďalej CA) je jednorozmerné pole buniek. (Môže byť konečné, jednosmerne alebo obojsmerne nekonečné.) Dvojrozmerný CA je dvojrozmerné pole buniek. Každá bunka je automat s konečným počtom stavov. Okolie bunky je množina jej susedov, ktorá má vplyv na stav bunky v ďalšom kroku. Čas je diskrétny a v každom kroku sa každá bunka nachádza v práve jednom zo stavov, ktorý sa mení podľa pravidiel každým krokom. Množina dvojíc buniek a ich stavov v konkrétnom čase sa nazýva konfigurácia. Počiatočnú konfiguráciu (v čase $t = 0$) tvorí nejaká konkrétna množina buniek a ich stavov. Každá ďalšia konfigurácia sa vypočíta z predchádzajúcej nasledovne:

1. pre každú bunku sa vypočíta jej nový stav, ktorý je určený súčasným stavom bunky a stavom jej susedov, táto funkcia sa tiež nazýva lokálne pravidlo alebo lokálna prechodová funkcia,

2. všetky bunky zmenia naraz svoj stav na nový, vypočítaný v prvom kroku.

V práci budeme skúmať bunkové automaty, ktoré majú tzv. *mŕtvy* stav. Tento stav je význačný tým, že bunka v mŕtvom stave môže prejsť do iného stavu (ožiť) len v prípade, ak sa v jej okolí nachádza aspoň jedna živá bunka. (Bunku budeme nazývať živou, ak nie je v mŕtvom stave.) Takýto typ CA je motivovaný skúmaním živých organizmov.

Veta 2.1.1. *Nech A je bunkový automat s mŕtvym stavom. Ak je v počiatočnej konfigurácii konečný počet živých buniek, tak v každom kroku sa v automate nachádza len konečný počet živých buniek.*

Dôkaz. Použijeme matematickú indukciu vzhľadom na čas t.

1° V čase $t = 0$ je počet živých buniek konečný.

2° Predpokladajme, že v čase $t = n$ je počet živých buniek konečný. Označme množinu živých buniek v čase n písmenom A a množinu mŕtvych susedov živých buniek v čase n písmenom B. Všimnime si, že živé bunky v čase $n + 1$ musia byť z jednej z množín A alebo B, pretože ostatné bunky sú v mŕtvom stave a nesusedia so žiadnou živou bunkou. Množina A je konečná z indukčného predpokladu. Množina B je tiež konečná, pretože každá živá bunka z množiny A má len konečný počet susedov, preto platí $|B| \leq |A| \cdot k$ pre k označujúce počet susedov bunky. □

Vďaka vete 2.1.1 sa bunkové automaty s mŕtvym stavom a konečnou počiatočnou konfiguráciou dajú simulovať na reálnom počítači.

2.1.1 Formálna definícia bunkového automatu

Definícia 2.1.1. Bunkový automat A definujeme ako päticu $A = ((L, \oplus), S, N, f, q_0)$, kde

- $(L, \oplus)$ je konečná alebo nekonečná grupa jednoznačne určujúca bunky v mriežke
- $S = \{q_0, q_1, \ldots, q_{k-1}\}$ je konečná množina stavov buniek
- $N = \{N_i\}_{i=0}^{m}$ je konečná postupnosť susedov taká, že $N_0 = e$, kde e je neutrálny prvok grupy $(L, \oplus)$
- $f : S^N \to S$ je lokálna prechodová funkcia
- q_0 je konštantný stav, t.j. platí $f(q_0, q_0, \ldots, q_0) = q_0$

Poznámka 2.1.1. Stav bunky $x \in L$ v čase t budeme označovať c_x^t. Množinu susedov bunky x označujeme ako n_x. Presnejšie $n_x = \{n_i^x\}_{i=0}^m$, kde $n_i^x = x \oplus N_i$.
Stav q_0 z definície budeme nazývať *mŕtvy*, ostatné stavy budeme nazývať živé. (Bunku v mŕtvom stave budeme nazývať mŕtva, bunku v živom stave budeme nazývať živá.)

Definícia 2.1.2. Konfigurácia C bunkového automatu $A = ((L, \oplus), S, N, f, q_0)$ je podmnožina $L \times (S \setminus \{q_0\})$, pričom platí, že ak $(x, p) \in C$ a $(x, q) \in C$, potom $p = q$. Je to množina dvojíc živých buniek a ich stavov.

Poznámka 2.1.2. Symbolom C^n budeme označovať n-tý krok výpočtu CA z počiatočnej konfigurácie C.

Definícia 2.1.3. Konfiguračná funkcia $F : 2^{L \times S \setminus \{q_0\}} \times L \to S$ je funkcia definovaná ako

$$F(C, x) = \begin{cases} q & \text{ak } (x, q) \in C, \\ q_0 & \text{ak } \forall q \quad (x, q) \notin C. \end{cases}$$

Stavy susedov bunky x v konfigurácii C budeme označovať C_{n_x}. To znamená, že $C_{n_x} = \{F(C, n_i^x)\}_{i=0}^m$.

Poznámka 2.1.3. Budeme používať operáciu $+$ na pričítanie prvku z L ku konfigurácii v nasledovnom význame. Ak $C = \{x_1, \ldots, x_n, \ldots\}$ tak $C + y = \{x_1 \oplus y, \ldots, x_n \oplus y, \ldots\}$.

Definícia 2.1.4. Krok výpočtu je relácia $\vdash$ na konfiguráciách definovaná nasledovne:

$$C^i \vdash C^{i+1} \iff f(C^i_{n_x}) = F(C^{i+1}, x) \quad \forall x \in L.$$

Poznámka 2.1.4. Lokálna prechodová funkcia teda mapuje stavy susedov bunky x do nasledujúceho stavu tejto bunky, čo môžeme zapísať ako $c_x^{t+1} = f(n_x^t)$.

Poznámka 2.1.5. V práci uvažujeme dvojrozmernú nekonečnú mriežku, ktorá je určená množinou $\mathbb{Z} \times \mathbb{Z}$ a operáciou $\oplus$, ktorá je definovaná po zložkách ako súčet v $\mathbb{Z}$.

2.2 Aplikácie CA

Štruktúra CA bola väčšinou študovaná na jedno- a dvoj-rozmerných nekonečných mriežkach. Za posledných 50 rokov boli CA dôsledne skúmané, boli využité ako základný model v rôznych odboroch vedy. Wolfram vo svojej knihe „A New Kind of Science“ [Wol59] vysvetľuje, prečo sú bunkové automaty rozšírené. Dôkazom toho môže byť aj fakt, že sa každý

rok objaví veľké množstvo nových publikácií. V ďalšom texte uvedieme stručný prehľad aplikácií bunkových automatov v rôznych vedných odvetviach, ktorý je čerpaný z článku [GSD+03]. Najskôr však čitateľa oboznámime so stručnou históriou bunkových automatov.

2.2.1 História

Bunkové automaty boli pôvodne navrhnuté Johnom von Neumannom ako formálny model samoreprodukujúcich sa organizmov. V období okolo roku 1950 von Neumann začal študovať mechanizmy, ktoré replikujú sami seba. Definoval univerzálny konštruktor. (Stroj, ktorý je schopný postaviť ďalší stroj podľa daného predpisu.) S Johnom von Neumannom pracoval aj Stan Ulam, ktorý vytvoril matematickú hru, ktorá vytvára nezvyčajné ale pôsobivé geometrické tvary. Rozdelil plochu na maticu bodov, ktoré môžu byť živé alebo mŕtve. Von Neumann vo svojom výskume použil nekonečnú maticu, ktorej bunky boli konečnostavové automaty. Po niekoľkých experimentoch definoval 29 stavov a prechodovú funkciu. Ukázal, že CA s danou konfiguráciou je schopný samoreprodukcie [lBoHMD98].

2.2.2 Vedecký výskum

V počítačových vedách sú CA základom von Neumannovho modelu samoreprodukcie. Boli tiež navrhnuté na budovanie paralelných násobiacich strojov [Col69], prvočíselných sít [Fis65], paralelných počítačov [Man77] a triediacich strojov [Nis81]. Našli svoje využitie aj v *fault-tolerant computing machine*(chybám odolných počítačoch) [Neu63]. Dvojrozmerné CA sú rozšírené pri spracovaní obrazu a pri rozlišovaní vzorov [Ros79].

CA sú rozšírené aj v oblasti biologických systémov, fyzikálnych modelov a chemických procesov. Fyzikálne javy sa zvyčajne opisujú pomocou diferenciálnych rovníc, ktoré sú však z praktického hľadiska často nepoužiteľné, a preto fyzici začali skúmať bunkové automaty, ktoré simulujú vlastnosti týchto javov. Ďalej sa používajú ako modely pre rôzne formy regulárneho, dendritického a náhodného rastu, založeného na dvojrozmerných CA [Pac86], pre formovanie vzorky v reakčno-difúznych systémoch [WWS85], na modelovanie hydrodynamických systémov [FHP86] atď. V ekológii sa CA používajú ako model predátor-korisť ekosystému [Res94], na odhaľovanie migrácie rýb v riekach [SK02] a model rastu zeleninovej populácie [BBK98]. Veľké množstvo aplikácií CA sa objavuje v súvislosti s DNA [BF84]. V chémii bunkové automaty reprezentujú modely chemických procesov ako absorpčno-desorpčného javu dôležitého pre analýzu jedov [CCF93], medzi-difúziu atómov dvoch materiálov [CDK89], zliatinové formovanie [Pac86] a tuhnúci proces so špeciálnou analýzou

stavu transformácie materiálu z tekutého na pevný [LG90].

Bunkové automaty sa využívajú nielen v prírodných vedách ale aj v sociálnych. James M. Sakoda bol prvý, kto vyvinul CA založené na modeli v sociálnych vedách. Thomas Schelling analyzoval odlučovací proces medzi jednotlivcami patriacimi do dvoch rozličných tried: čierny a biely [Sch71]. V poslednom desaťročí sa modely založené na CA používajú oveľa častejšie v behaviorálnych a sociálnych vedách. V ekonomike sa jednorozmerný CA používa na modelovanie a analyzovanie oceňovania v priestorovom stave [KO93]. Slúži aj ako model efektu ekonomickej nerovnomernosti v objavujúcom sa trhu [GM02], dopravného prúdu [GW95] takisto ako dizajnový nástroj pre mestský rozvoj [RU02].

2.3 Parametre v definícii bunkových automatov

Existuje viacero aspektov, ktoré ovplyvňujú silu bunkových automatov. Sú to napríklad typ bunky a s tým spojený typ okolia a jeho veľkosť, pravidlá alebo počet stavov. V tejto sekcii popíšeme niektoré z nich podrobnejšie.

2.3.1 Okolia

Dôležitým faktorom pri analyzovaní vlastností bunkových automatov je bezpochyby typ okolia, ktorým sú medzi sebou bunky poprepájané. Dve najčastejšie voľby výberu susedov pre bunkové automaty na štvorčekovej mriežke sú von Neumannove okolie a okolie podľa Moorea. (Tieto okolia sú definované len na štvorčekovej mriežke.) Výsledky spojené s týmito okoliami, ktoré popisujeme nižšie, môže čitateľ nájsť v [Kar05].

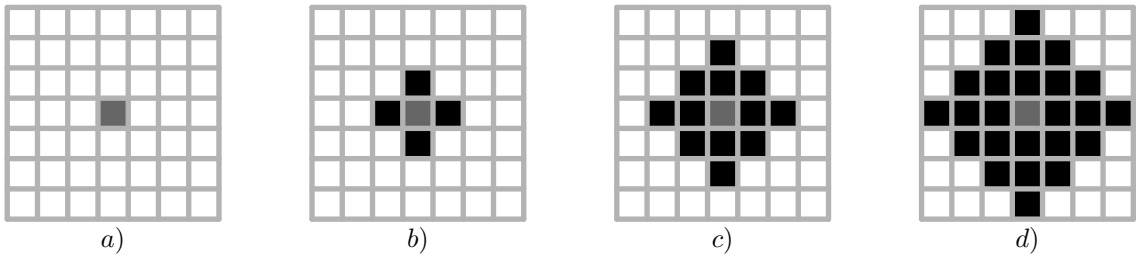
Von Neumannovo okolie

Von Neumannovo okolie je okolie v tvare diamantu. Množina buniek, ktoré sú susedmi danej bunky (x_0, y_0) pri danej vzdialenosti r, je definovaná ako

$$N^{\mathcal{V}}_{(x_0,y_0,r)} = \{(x, y) \ : \ |x - x_0| + |y - y_0| \leq r\}.$$

Počet buniek v okolí danej bunky pri vzdialenosti r je $2 \cdot r(r + 1) + 1$. Pre $r = 0, 1, 2, 3 \ldots$ je počet buniek v okolí postupne $1, 5, 13, 25 \ldots$[Weib]. (Viď obrázok 2.1.)

V literatúre sa pod von Neumannovým okolím niekedy myslí okolie so vzdialenosťou $r = 1$. V tomto prípade sa v okolí bunky podľa von Neumanna nachádzajú bunky napravo, naľavo, nad a pod touto bunkou, ako aj bunka sama. (Viď obrázok 2.1b.)



Obr. 2.1: Von Neumannove okolie so vzdialenosťou *a)* $r = 0$ *b)* $r = 1$ *c)* $r = 2$ *d)* $r = 3$

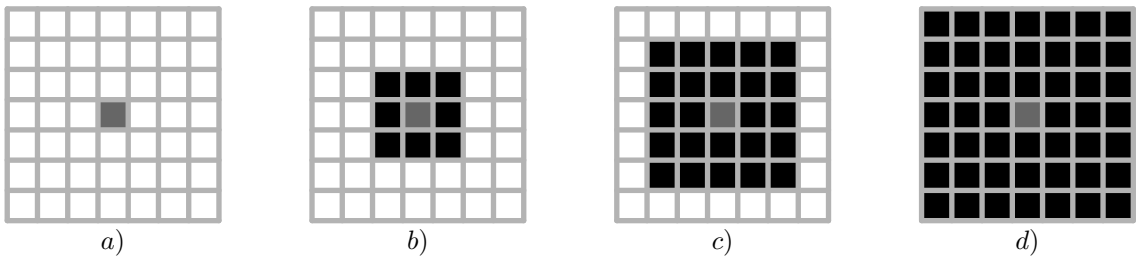
Moorove okolie

Druhým často používaným typom okolia je Moorove okolie. Toto okolie má tvar štvorca, množina susedov danej bunky (x_0, y_0) so vzdialenosťou r je

$$N^{\mathcal{M}}_{(x_0,y_0,r)} = \{(x, y) : |x - x_0| \leq r, |y - y_0| \leq r\}.$$

Počet susedov danej bunky je $(2 \cdot r + 1)^2$. Pre $r = 0, 1, 2, 3 \ldots$ je to postupne $1, 9, 25, 49 \ldots$ [Weia] (Viď obrázok 2.2.)

Niekedy sa pod Mooreovým okolím myslí okolie so vzdialenosťou $r = 1$, kedy má bunka za susedov všetky bunky, s ktorými má spoločnú hranu alebo vrchol. (Viď obrázok 2.2b.)



Obr. 2.2: Mooreove okolie so vzdialenosťou *a)* $r = 0$ *b)* $r = 1$ *c)* $r = 2$ *d)* $r = 3$

Okolia na šesťuholníkovej mriežke

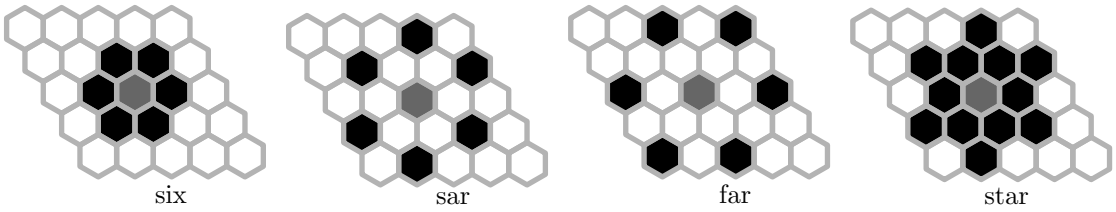
V rámci práce pracujeme so šesťuholníkovými bunkovými automatmi. Tieto automaty pracujú formálne nad rovnakou mriežkou ako štvorcové automaty (t.j. nad $\mathbb{Z} \times \mathbb{Z}$), no líšia sa okolím. Medzi najčastejšie typy okolí pre šesťuholníkové automaty patria okolia z obrázku 2.3 [May03]. My budeme používať okolie *six* ktoré je dané postupnosťou $N = ((0, 0), (0, -1), (1, -1), (1, 0), (0, 1), (-1, 1), (-1, 0))$.

Postupnosti susedov pre zvyšné typy okolí sú:

sar $N = ((0,0),(1,-2),(2,-1),(1,1),(-1,2),(-2,1),(-1,-1))$

far $N = ((0,0),(0,-2),(2,-2),(2,0),(0,2),(-2,2),(-2,0))$

star $N = ((0,0),(0,-1),(1,-1),(1,0),(0,1),(-1,1),(-1,0),(1,-2),(2,-1),(1,1),$
$(-1,2),(-2,1),(-1,-1))$



Obr. 2.3: Typy okolí v šesťuholníkovej sieti

2.3.2 Pravidlá

V teórii bunkových automatov sa často predpokladá, že lokálne pravidlá sú deterministické, no nie je to nutné. Existuje aj nedeterministické mapovanie, ktoré sa študuje v súvislosti s rečovou teóriou. Bunkové automaty, kde má každá bunka svoje vlastné lokálne pravidlo, sa nazývajú hybridné. Existujú aj bunkové automaty, v ktorých sa lokálne pravidlo pre danú bunku mení každým krokom. Tieto CA sa nazývajú programovateľné alebo tiež mozaikové [Sar00]. Ako vidieť z definície bunkových automatov, my sa zaoberáme takými CA, kde je lokálne pravidlo deterministické, pre každú bunku rovnaké a nemenné.

Zápis pravidiel

V nasledujúcom texte uvedieme dva často využívané zápisy pravidiel v bunkových automatoch. Prvý z nich, Wolframov zápis, uvedieme pre jednoduchosť na jednorozmerných bunkových automatoch.

Wolfram skúmal a klasifikoval jednorozmerné CA s dvoma susedmi. (Napravo a naľavo od bunky. Automaty s takýmto typom okolia a dvoma stavmi sa nazývajú základné.) Navrhol schému, ktorá sa stala štandardom. Každé základné pravidlo v tejto schéme je špecifikované sekvenciou ôsmich bitov.

$$f(111)\ f(110)\ f(101)\ f(100)\ f(011)\ f(010)\ f(001)\ f(000)$$

Sekvencia bitov tvorí binárny zápis celého čísla z intervalu 0—255. Toto číslo sa tiež nazýva Wolframovo číslo CA.

Zápis $f(abc) = d$ znamená, že ak má ľavý sused stav a, daná bunka stav b a pravý sused stav c, tak daná bunka bude mať v ďalšom kroku stav d. Napríklad slávne pravidlo 110 je bunkový automat, kde

$$f(111) = 0 \quad f(110) = 1 \quad f(101) = 1 \quad f(100) = 0$$
$$f(011) = 1 \quad f(010) = 1 \quad f(001) = 1 \quad f(000) = 0$$

získané z binárneho zápisu $110 = (01101110)_2$.

V týchto pravidlách záleží na umiestnení živých buniek okolo danej bunky. Iný zápis pravidiel je napríklad taký, že záleží iba na počte živých buniek v okolí danej bunky. (Na rozložení nezáleží.) Tieto pravidlá budeme zapisovať ako dvojicu čísel, ktoré budú oddelené znakom „/“. Napr. 23/4 znamená, že ak má živá bunka dvoch alebo troch živých susedov tak prežije do ďalšieho kroku. Ak má mŕtva bunka práve štyroch živých susedov tak v ďalšom kroku z nej bude živá bunka.

Klasifikácia pravidiel

Wolfram experimentoval s pravidlami CA začínajúc z náhodnej počiatočnej konfigurácie, ktoré boli založené na pozorovaní na priestorovo-časových diagramoch a klasifikoval pravidlá do štyroch kategórií:

($W1$) väčšina počiatočných konfigurácií vedie k nejakej fixnej bodovej konfigurácii

($W2$) väčšina počiatočných konfigurácií vedie k periodicky sa opakujúcej konfigurácii

($W3$) väčšina počiatočných konfigurácií vedie k v podstate náhodne vyzerajúcemu pozadiu

($W4$) lokalizovaná štruktúra s komplexnou objavujúcou sa interakciou

Wolfram sa domnieval, že 4. trieda CA je výpočtovo univerzálna. Klasifikácia podľa Wolframa je však nepresná a neskôr bola formalizovaná Culikom a Yuom, ktorí dokázali, že ich klasifikácia je nerozhodnuteľná. (Neexistuje algoritmus, ktorý by určil, do ktorej triedy patrí daný CA.) Uvažujme CA s mŕtvym stavom q. Triedy Culik-Yu sú definované podľa nasledovných vlastností. Bunkový automat patrí do najnižšej triedy, ktorej definičná vlastnosť je splnená.

($CY1$) všetky konečné konfigurácie sa vyvinú do mŕtvej konfigurácie $Q = \emptyset$, t.j. pre každú konečnú konfiguráciu $C \subset L \times S \setminus \{q\}$ platí $C \vdash^* Q$

($CY2$) všetky konečné konfigurácie sú nakoniec periodické, t.j. pre každú konečnú konfiguráciu $C \subset L \times S \setminus \{q\}$, kde C je konečná, existujú $m \geq 0$ a $n \geq 1$ také, že $C \vdash^m C' \vdash^n C'$

($CY3$) existuje algoritmus, ktorý zistí, či daná konečná konfigurácia je dosiahnuteľná z inej danej konečnej konfigurácie t.j. je rozhodnuteľné pre dané konečné konfigurácie $C, E \subseteq L \times S \setminus \{q\}$, či $C \vdash^* E$

($CY4$) žiadne obmedzenie

Tieto a iné zaujímavosti spojené so základnými CA môže čitateľ nájsť v [Kar05].

Kapitola 3

Game of Life

Bunkové automaty sa tiež využívajú ako model v mnohých hrách. Najznámejšia sa volá Game of Life a bola navrhnutá Johnom Conwayom a jeho kolegami v roku 1970. V tejto kapitole popíšeme túto hru, formálne definujeme zodpovedajúci bunkový automat a ukážeme niekoľko najznámejších vzorov (konfigurácií), ktoré sa v hre vyskytujú.

3.1 Popis hry

Game of Life sa hrá na nekonečnej dvojrozmernej štvorčekovej mriežke, ktorej každý štvorček tvorí jednu bunku t.j. jeden automat s dvomi stavmi: živý alebo mŕtvy. Každá bunka má 8 susedov, štyroch na diagonálach a štyroch na ortogonálach. Na začiatku hry si hráč zvolí počiatočnú konfiguráciu (vzor) a potom už len sleduje ako sa vzor mení na základe pravidiel[1] pre narodenie, smrť a prežitie. Conway vyberal tieto pravidlá veľmi pozorne a s početným experimentovaním, aby spĺňali tieto kritériá:

($C1$) Nemal by byť žiaden počiatočný vzor, pre ktorý existuje jednoduchý dôkaz, že počet živých buniek môže rásť bez obmedzenia

($C2$) Mal by byť počiatočný vzor, ktorý zdanlivo rastie bez obmedzenia

($C3$) Mal by byť jednoduchý počiatočný vzor, ktorý rastie a mení sa vo významnej perióde predtým než zahynie v troch možných cestách: uniká celý preč, usadí sa v stabilnej konfigurácii, ktorá zostáva naďalej nezmenená alebo prejde do oscilačnej fázy kde opakuje nekonečný cyklus o perióde dva alebo viac

[1] Formálne sa použije lokálna prechodová funkcia CA

Pravidlá, ktorými sa riadia všetky bunky sú veľmi jednoduché.

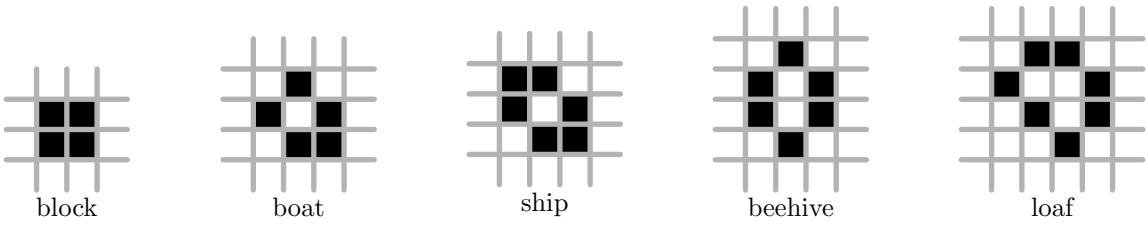
- Prežívajúci. Každá živá bunka s dvomi alebo tromi živými susedmi prežije do ďalšej generácie.
- Umierajúci. Každá živá bunka so štyrmi alebo viacerými živými susedmi zomrie. Každá živá bunka s jedným alebo žiadnym živým susedom zomrie.
- Narodený. Každá mŕtva bunka s práve tromi živými susedmi ožije.

Tieto pravidlá môžeme tiež zapísať ako 23/3.

3.2 Typy vzorov v Game of Life

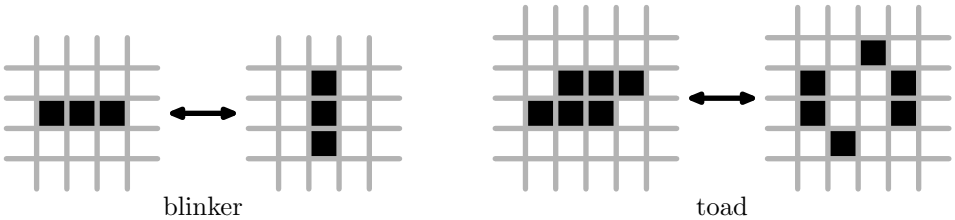
Experimentovaním s počiatočnými konfiguráciami sa zistilo, že existujú skupiny vzorov správajúce sa podľa určitých pravidiel.

„Still life“ sú nehybné vzory, ktoré zostávajú rovnaké každý krok. Žiadna živá bunka nezomrie a žiadna bunka sa nenarodí. Pre každý vzor, ktorý je „still life“ platí, že každá živá bunka musí mať 2 alebo 3 živých susedov a každá mŕtva bunka má počet živých susedov rôzny ako 3. Najznámejšie vzory sú na obrázku 3.1.



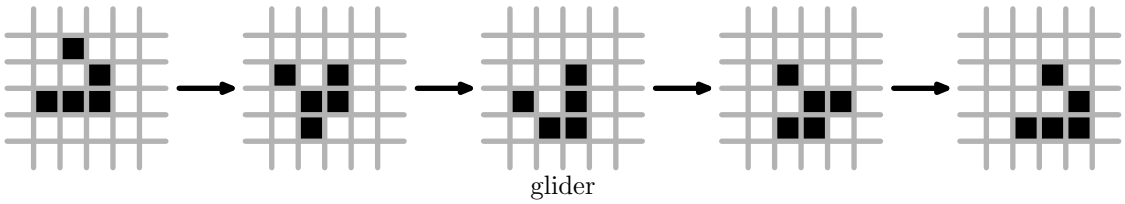
Obr. 3.1: Nehybné (still life) vzory

„Oscillators“ sú oscilujúce vzory, ktoré sa periodicky menia v každom kroku. Dva najjednoduchšie oscilátory môžeme vidieť na obrázku 3.2, obidva majú periódu 2.



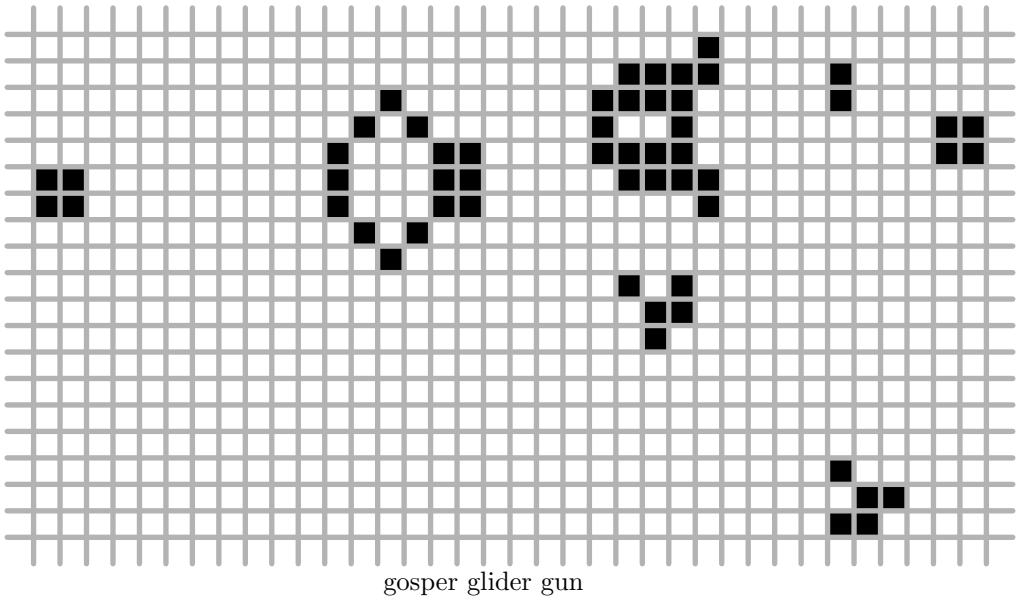
Obr. 3.2: Oscilátory

„Spaceship“ alebo tiež vesmírne lode sú vzory pohybujúce sa po mriežke. Na obrázku 3.3 môžeme vidieť jeden z najjednoduchších pohybujúcich sa vzorov, glider.



Obr. 3.3: Vesmírne lode

„Glider guns“ sa od osciátorov líšia tým, že počas každej periódy vypúšťajú jeden alebo viac vesmírnych lodí. (Viď. obrázok 3.4)



Obr. 3.4: Oscilátory vypúšťajúce vesmírne lode

3.3 Formálny zápis Game of Life

Definícia 3.3.1. Bunkový automat Game of Life je pätica $((L, \oplus), S, N, f, 0)$, kde

- L je $\mathbb{Z} \times \mathbb{Z}$ alebo $\mathbb{Z}_n \times \mathbb{Z}_n$ v závislosti od toho, či uvažujeme konečnú alebo nekonečnú mriežku a operácia $\oplus$ je definovaná po zložkách ako súčet v $\mathbb{Z}$ alebo $\mathbb{Z}_n$

- $S = \{0, 1\}$ je množina stavov buniek
- $N = ((0,0),(-1,1),(0,1),(1,1),(-1,0),(1,0),(-1,-1),(0,-1),(1,-1))$ je postupnosť susedov
- f je lokálna prechodová funkcia definovaná nasledovne:

$$f(a_0, a_1, \ldots, a_m) = \begin{cases} 1 & \text{ak } \sum_{k=1}^{m} a_k = 3 \wedge a_0 = 0 \\ 1 & \text{ak } \sum_{k=1}^{m} a_k \in \{2,3\} \wedge a_0 = 1 \\ 0 & \text{inak} \end{cases}$$

Príklad 3.3.1. Výpočet pre konfiguráciu glider z obrázku 3.3 môžme zapísať nasledovne:

$$\begin{aligned} C_0 = \ & \{(1,3),(2,1),(2,3),(3,2),(3,3)\} \vdash \{(1,2),(2,3),(2,4),(3,2),(3,3)\} \vdash \\ \vdash \ & \{(1,3),(2,4),(3,2),(3,3),(3,4)\} \vdash \{(2,2),(2,4),(3,3),(3,4),(4,3)\} \vdash \\ \vdash \ & \{(2,4),(3,2),(3,4),(4,3),(4,4)\} = C_4 \end{aligned}$$

Môžeme si všimnúť, že platí $C_4 = C_0 + (1,1)$.

3.4 Formálny zápis typov vzorov v Game of Life

Keď už máme formálnu definíciu hry Game of Life, môžeme definovať aj typy vzorov spomenuté vyššie. Pod vzorom rozumieme počiatočnú konfiguráciu.

„Nehybné vzory“ $\forall t \geq 0$ platí, že $C^t = C^{t+1}$

„Oscilátory“ $\forall t \geq 0$ platí, že $C^t = C^{t \bmod k}$, kde $k > 1$ je perióda oscilátora

„Vesmírne lode“ $\forall t \geq 0$ platí, že $C^t = C^{t \bmod k} + (\lfloor t/k \rfloor \cdot \vec{u})$, kde $k > 1$ je perióda oscilátora a $\vec{u} \neq (0,0)$ je posun vesmírnej lode po prejdení jednej periódy

„Oscilátory vypúšťajúce vesmírne lode“ $\forall t \geq 0$ platí, že $C^t = C^{t \bmod k} \cup (\bigcup_{i=0}^{a_t} L_i)$, kde $k > 1$ je perióda oscilátora, množiny $C^{t \bmod k}, L_0, \ldots, L_{a_t}$ sú navzájom disjunktné a majú vo svojom okolí len mŕtve bunky. Vzory $L_0, \ldots, L_{a_t}$ sú vesmírne lode a $\lim_{j \to \infty} a_j = \infty$.

Kapitola 4

Program simulujúci šesťuholníkové bunkové automaty

Súčasťou bakalárskej práce je aj program, ktorý simuluje bunkové automaty na šesťuholníkovej mriežke. Jedným z dôvodov, prečo máme vlastnú implementáciu bunkových automatov, je väčšia flexibilita pri experimentálnom skúmaní CA. V tejto kapitole analyzujeme zložitosť programu a stručne rozoberáme možnosti voľby dátovej štruktúry na reprezentáciu konfigurácie.

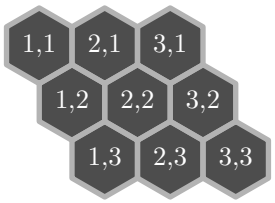
4.1 Analýza programu

Program na simuláciu bunkového automatu je naprogramovaný v jazyku `C++`. Použili sme dva základné objekty: bunku a pravidlá.

Bunka má stavové premenné x-ovú a y-ovú súradnicu, ktoré udávajú jej pozíciu v šesťuholníkovej mriežke, tak ako je zobrazené na obrázku 4.1. Súradnice obmedzíme veľkosťou premennej typu `long`. Takýto rozmer mriežky považujeme pre naše účely za dostatočný. Keďže v programe ukladáme živé bunky do množiny, zaviedli sme na bunkách totálne usporiadanie definované nasledovne. Bunka so súradnicami (x_1, y_1) je menšia ako bunka so súradnicami (x_2, y_2) práve vtedy, keď platí, že

$$x_1 < x_2 \vee (x_1 = x_2 \wedge y_1 < y_2)$$

V rámci objektu pravidlá používame dve polia, ktoré určujú sadu pravidiel. Tieto pravidlá sa načítavajú pri inicializácii zo súboru. V jednom z polí (`zz`) sú uložené počty živých



Obr. 4.1: Šesťuholníkové bunky v súradnicovej sústave

susedov živej bunky, pre ktoré táto bunka prežije. V druhom poli (`mz`) sú uložené počty živých susedov mŕtvej bunky, pre ktoré táto bunka ožije. Objekt pravidlá má jednu metódu `vykonaj`, ktorá spočíta počet živých a mŕtvych susedov danej bunky a na základe aktuálneho stavu bunky a pravidiel určí, či daná bunka bude mať v nasledujúcom kroku stav živá alebo mŕtva.

Na vykreslenie mriežky sme použili knižnicu OpenGL. Najdôležitejšia časť programu je spôsob pamätania si živých buniek v mriežke a zisťovanie, ktoré bunky do nasledujúceho kroku ožijú. Pre ďalšiu analýzu označme znakom n rozmery mriežky bunkového automatu.

Na zložitosť programu má vplyv viacero aspektov. Od typu okolia, resp. počtu buniek v okolí, závisí veľkosť množiny *kandidátov* na živé bunky v ďalšom kroku. Táto množina je samozrejme závislá aj od počtu živých buniek. V množine kandidátov môžu byť len živé bunky a bunky v okolí živých buniek, pretože mŕtva bunka, ktorá má všetky bunky v okolí mŕtve, nemôže ožiť. V každom kroku potrebujeme kontrolovať, ktorí kandidáti naozaj ožijú/prežijú. Táto kontrola pre jedného kandidáta závisí od okolia a od voľby dátovej štruktúry.

4.1.1 Pole

Jednou z najjednoduchších dátových štruktúr, ktorú môžeme použiť na pamätanie si živých buniek v mriežke, je dvojrozmerné pole. Výhodou takéhoto riešenia je konštantná časová zložitosť pri zisťovaní, či je bunka živá alebo mŕtva. Z toho vyplýva, že aj zisťovanie stavu v nasledujúcom kroku pre danú bunku zaberá konštantný čas. Ak však chceme prechádzať bunky, ktoré majú možnosť ožiť/prežiť do ďalšieho kroku, musíme prechádzať celé pole. Preto prechod bunkového automatu do nasledujúceho stavu bude trvať $\mathcal{O}(n^2)$. Ďalšou nevýhodou tohto riešenia je pamäťová zložitosť. Keďže si pamätáme stavy všetkých buniek, je pamäťová zložitosť $\mathcal{O}(n^2)$.

4.1.2 Množina a hašovacia množina

Keďže program potrebuje často prechádzať všetky živé bunky a mŕtve bunky, ktoré môžu ožiť do nasledujúceho kroku, bolo by vhodné, keby sme používali takú štruktúru, ktorá by vedela rýchlo vyhľadávať a prechádzať všetky prvky. Pri implementácii sme vybrali dve štruktúry, ktoré zodpovedajú našim požiadavkám: množinu (`set` z [SGI97]) a hašovaciu množinu (`google::dense_hash_set` z [Goo05]).

V programe používame tri množiny (hašovacie množiny). V jednej z nich máme uložené aktuálne živé bunky. Pri výpočte nasledujúceho kroku vytvoríme druhú množinu, ktorá obsahuje všetkých mŕtvych susedov živých buniek. (Len tieto bunky môžu v nasledujúcom kroku ožiť.) Potom pre všetky živé bunky a všetky bunky susedov zavoláme metódu `vykonaj`. Ak má byť bunka živá v nasledujúcom kroku, uložíme ju do tretej množiny. Po ukončení tohto výpočtu bude tretia množina považovaná za aktuálne živé bunky.

Z obidvoma spomenutými štruktúrami sa z užívateľského hľadiska pracuje rovnako – obe obsahujú metódy na nájdenie, vloženie a vymazanie bunky z množiny. Vytvorili sme dve alternatívy programu, ktoré sa odlišujú len použitou štruktúrou. Pri väčších počtoch živých buniek pracuje rýchlejšie program, ktorý používa hašovaciu množinu, pri menších počtoch pracuje rýchlejšie program používajúci množinu.

Množina

Štruktúra množina si pamätá prvky v utriedenom poradí. Z toho dôvodu môže vyhľadávať prvky binárnym prehľadávaním, a teda má časovú zložitosť vyhľadávania $\mathcal{O}(\log m)$, kde m je počet buniek uložených v množine.

Hašovacia množina

Druhá štruktúra, hašovacia množina, má časovú zložitosť vyhľadávania v priemernom prípade konštantnú. Vybrali sme implementáciu `google::dense_hash_set`, ktorá pracuje rýchlejšie ako iné implementácie hašovacej množiny za cenu väčšej pamäťovej náročnosti.

4.2 Časová zložitosť

Uvedieme časovú zložitosť jedného kroku algoritmu. Označme počet živých buniek ako n a veľkosť okolia m. Pri kroku výpočtu najskôr skonštruujeme množinu mŕtvych kandidátov,

potom aplikujeme pravidlá na živé bunky a mŕtvych kandidátov, čím dostaneme množinu živých buniek v nasledujúcom kroku.

Množina mŕtvych kandidátov obsahuje maximálne mn buniek. (Každá bunka, ktorá je susedom živej bunky môže byť kandidát, preto sa pri každej pozrieme, či je naozaj mŕtva.) Táto konštrukcia trvá $\mathcal{O}(mn \log n)$ pre štruktúru `set` a $\mathcal{O}(mn)$ pre štruktúru `dense_hash_set`.

Pre každú bunku, ktorá môže byť živá v ďalšom kroku, zistíme aký stav majú susedia tejto bunky. (Nájdeme ju v poli živých buniek.) Celkový počet buniek, ktoré môžu ožiť, nie je väčší ako $mn + n$. Zložitosť tohto výpočtu je $\mathcal{O}(m^2 n \log n)$ pre štruktúru `set` a $\mathcal{O}(m^2 n)$ pre štruktúru `dense_hash_set`.

Kapitola 5

Požiadavky pre pravidlá Game of Life

V rámci študovania materiálov ku bakalárskej práci, sme našli údajne jediné pravidlo (3/2), ktoré je ekvivalentom ku Conwayovmu Game of Life na šesťuholníkovej mriežke [Bay02]. Toto tvrdenie však nie je nikde zdôvodnené. V ďalšom texte bude preto našim cieľom preskúmať všetky pravidlá a vybrať tie, ktoré najviac vyhovujú Conwayovým podmienkam.

Budeme pracovať s rovnako jednoduchými pravidlami ako pracoval Conway, t.j. nebude nás zaujímať poloha živých buniek v okolí, ale len ich počet. V tejto kapitole budeme vylučovať nevhodné pravidlá, ktoré nevyhovujú Conwayovým podmienkam pre Game of Life. Na záver uvedieme výsledky testovania vhodných pravidiel na náhodných vstupoch.

5.1 Conwayove kritériá

Vhodné pravidlá pre šesťuholníkovú verziu Game of Life budeme vyberať podľa Conwayových kritérií spomenutých v Kapitole 3. Vzhľadom k ďalšiemu použitiu ich uvádzame ešte raz.

($C1$) Nemal by byť žiaden počiatočný vzor, pre ktorý existuje jednoduchý dôkaz, že počet živých buniek môže rásť bez obmedzenia

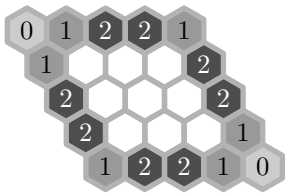
($C2$) Mal by byť počiatočný vzor, ktorý zdanlivo rastie bez obmedzenia

($C3$) Mal by byť jednoduchý počiatočný vzor, ktorý rastie a mení sa vo významnej perióde predtým než zahynie v troch možných cestách: uniká celý preč, usadí sa v stabilnej konfigurácii, ktorá zostáva naďalej nezmenená alebo prejde do oscilačnej fázy kde opakuje nekonečný cyklus o perióde dva alebo viac

Celkový počet pravidiel, ktoré budeme skúmať je $(2^6 - 1)^2 = 3969$. V nasledujúcich sekciách vylúčime tie z nich, ktoré nemôžu vyhovovať niektorému z Conwayových kritérií.

5.2 Druhé kritérium

Na základe druhého kritéria sa pokúsime vylúčiť niektoré nezodpovedajúce pravidlá. Druhá podmienka uvádza, že by mala existovať počiatočná konfigurácia, ktorá zdanlivo rastie bez obmedzenia.



Obr. 5.1: Počet susedov z kosoštvorcovej vnútornej oblasti

Veta 5.2.1. *Ak má platiť druhé Conwayove kritérium, tak v pravidlách na oživenie bunky musí byť pravidlo s číslom 1 alebo 2.*

Dôkaz. Ak má konfigurácia rásť bez obmedzenia, musia po niekoľkých krokoch živé bunky prekročiť danú hranicu ľubovoľnej veľkosti. Uvažujme hranicu v tvare kosoštvorca. (Viď obrázok 5.1.) Predpokladáme teda, že všetky živé bunky majú súradnice (a, b), kde $0 \leq a, b \leq n$. Preskúmajme bunky, ktoré tvoria hranicu oblasti a pôsobenie buniek vo vnútri ohraničeného územia na ne. Bunky označené číslom 0 nemajú vo vnútri žiadneho suseda, bunky s číslom 1 majú jedného suseda a bunky s číslom 2 majú dvoch susedov. Ak sa chceme dostať cez tieto hranice, musíme mať v pravidlách na oživenie bunky pravidlo s číslom 1 alebo 2. (V opačnom prípade, ak sú všetky živé bunky v ohraničenej oblasti, bunky na hranici – a teda aj za hranicou – nemajú ako ožiť.) Pravidlo s číslom 0 nemôžeme mať, lebo bunka nemôže ožiť, ak nemá v okolí živého suseda. □

5.3 Prvé kritérium

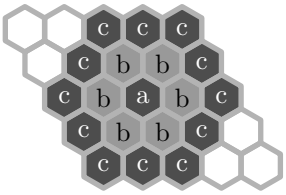
Venujme sa teraz prvej Conwayovej podmienke. Keďže pojem „počiatočný vzor, ktorý rastie bez obmedzenia“ nebol jednoznačný, pre naše účely ho definujeme nasledovne.

Definícia 5.3.1. Budeme hovoriť, že počiatočná konfigurácia C silne rastie bez obmedzenia, ak

$$\lim_{n\to\infty} |C^n| = \infty$$

Veta 5.3.1. *Pre každé pravidlo tvaru $*/12*$ existuje jednoduchá počiatočná konfigurácia, ktorá silne rastie bez obmedzenia.*

Dôkaz. Pre vzor na obrázku 5.2 vieme jednoducho dokázať, že silne rastie bez obmedzenia. V každom ďalšom kroku ožijú všetky bunky, ktoré susedia so živými bunkami. (Niektoré živé bunky môžu umrieť.) Pri začiatočnej konfigurácii s jedinou živou bunkou (na obrázku označená písmenom a), v prvom kroku ožije 6 buniek (označených písmenom b), v druhom kroku 12 buniek (písmeno c). Všeobecne v n-tom kroku ožije aspoň $(n+1)\cdot 6 - 6$ buniek, čo znamená, že pre ľubovoľné k vieme nájsť n také, že v n-tom kroku bude živých aspoň k buniek. Z toho vyplýva, že počet živých buniek silne rastie bez obmedzenia. □



Obr. 5.2: Ožívajúce bunky pre pravidlá $*/12*$

Príklad 5.3.1. Uvažujme pravidlo 1/1 a počiatočnú konfiguráciu $C = \{(0,0)\}$. Táto konfigurácia sa po dvoch krokoch dostane do konfigurácie

$$C^2 = \{(0,-2),(2,-2),(2,0),(0,2),(-2,-2),(-2,0)\}.$$

Všeobecne po 2^n krokoch pre $n \geq 1$ bude konfigurácia

$$C^{2^n} = \{(0,-2^n),(2^n,-2^n),(2^n,0),(0,2^n),(-2^n,-2^n),(-2^n,0)\}.$$

Môžeme si všimnúť, že $\lim_{n\to\infty} |C^n|$ neexistuje, preto konfigurácia C nerastie silne bez obmedzenia podľa našej definície. Napriek tomu každá konfigurácia C^{2^n-1} má $3\cdot 2^{2n-1}$ buniek. Z toho je nám intuitívne jasné, že takáto konfigurácia rastie bez obmedzenia.

Príklad 5.3.1 ukazuje, že naša definícia pojmu *konfigurácia rastúca bez obmedzenia* nie je správna. Vyslovíme preto upravenú definíciu, ktorá bude spĺňať naše požiadavky.

Definícia 5.3.2. Budeme hovoriť, že počiatočná konfigurácia C rastie bez obmedzenia, ak

$$\limsup_{n\to\infty} |C^n| = \infty$$

Poznámka 5.3.1. Čitateľ si môže všimnúť, že takáto voľba definície nezmení dôkaz vety 5.3.1.

Veta 5.3.2. *Pre každé pravidlo tvaru $*/1*$ existuje jednoduchá počiatočná konfigurácia, ktorá rastie bez obmedzenia.*

Dôkaz. Ukážeme, že počiatočná konfigurácia $C = \{(0,0)\}$ rastie bez obmedzenia pre ľubovoľné pravidlo tvaru $*/1*$. Najprv indukciou dokážeme, že v n-tom kroku sú bunky $(0,-n),(n,-n),(n,0),(0,n),(-n,n),(-n,0)$ živé a zároveň všetky živé bunky sú v šesťuholníku so stranou dĺžky $n+1$ so stredom v $(0,0)$. (Pre $n \geq 1$.)

1° Všimnime si, že konfigurácia po prvom kroku spĺňa obidve podmienky tvrdenia, pretože $C^1 = \{(0,-1),(1,-1),(1,0),(0,1),(-1,1),(-1,0)\}$.

2° Nech v n-tom kroku žijú bunky $(0,-n),(n,-n),(n,0),(0,n),(-n,n),(-n,0)$ a zároveň všetky živé bunky sú v šesťuholníku so stranou dĺžky $n+1$ so stredom v $(0,0)$. Potom v $(n+1)$-om kroku ožijú bunky

$$(0,-n-1),(n+1,-n-1),(n+1,0),(0,n+1),(-n-1,n+1),(-n-1,0),$$

pretože majú práve jedného živého suseda. (Každá jednu zo živých buniek z indukčného predpokladu.) Keďže v n-tom kroku boli všetky živé bunky v šesťuholníku so stranou dĺžky $n+1$, všetky bunky, ktoré môžu ožiť v $(n+1)$-om kroku sú v šesťuholníku so stranou dĺžky $n+2$. (Každá mŕtva bunka, ktorá má ožiť, musí susediť s nejakou živou bunkou.)

Môžeme si všimnúť, že hranica šesťuholníkovej oblasti sa každým krokom zväčšuje, navyše to, ktoré bunky budú živé na tejto hranici v $(n+1)$-om kroku, záleží len na bunkách na predchádzajúcej hranici v n-tom kroku. Predpokladajme, že v pravidlách na oživenie nie je číslo 2. (Opačný prípad – pravidlá tvaru $*/12*$ – sme vyriešili vo vete 5.3.1.) Keďže bunky na hranici majú jedného alebo dvoch susedov vo vnútornej oblasti a v pravidlách na oživenie sa nachádza číslo 1, pozície živých buniek na hraniciach sú dané jednoznačne.

Pozrime sa na jednu stranu hraničného šesťuholníka, t.j. na bunky $(0,-n),\ldots,(n,-n)$. Indukciou ukážeme, že počet živých buniek na tejto strane v kroku 2^n-1 je 2^n. Z toho

vidieť, že počet živých buniek v kroku $2^n - 1$ je väčší alebo rovný 2^n, z čoho vyplýva, že

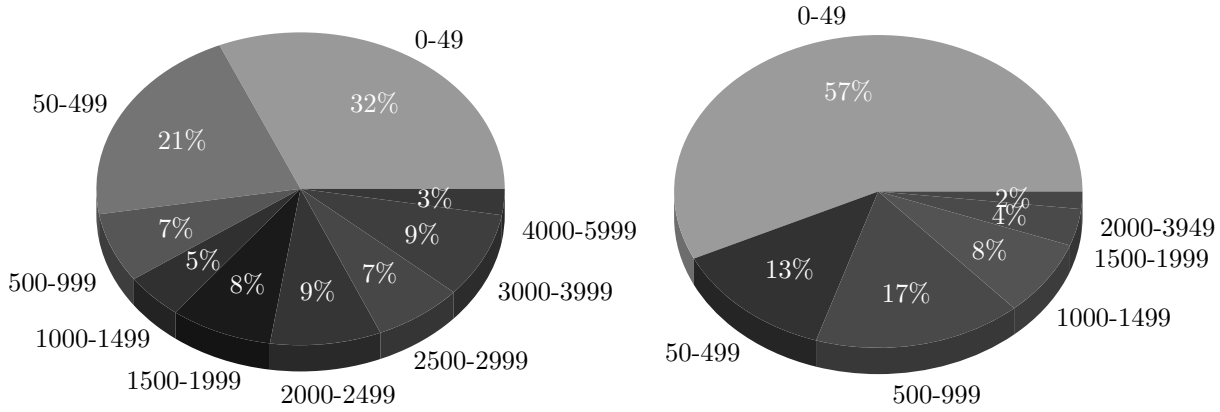
$$\limsup_{n\to\infty} |C^n| = \infty.$$

Predtým ako uvedieme dôkaz predchádzajúceho tvrdenia si treba uvedomiť, že ak v kroku $2^n - 1$ žije na strane 2^n buniek, tak to znamená, že žijú všetky bunky na tejto strane.

1° V kroku $2^1 - 1 = 1$ sú živé bunky $(0, -1), (1, -1)$. ich počet je 2^1.

2° Nech sú v kroku $2^n - 1$ živé všetky bunky $(0, -(2^n - 1)), \ldots, (2^n - 1, -(2^n - 1))$. Z tvaru pravidiel vyplýva, že v kroku 2^n budú na jednej strane živé len tieto dve bunky: $(0, -2^n), (2^n, -2^n)$. Tieto dve bunky budú ovplyvňovať hranicu len „okolo seba“, preto po ďalších $2^n - 1$ krokoch vznikne z bunky $(0, -2^n)$ rad buniek $(0, -2^{n+1} + 1), \ldots, (2^n - 1, -2^{n+1} + 1)$ a z bunky $(2^n, -2^n)$ vznikne rad $(2^n, -2^{n+1} + 1), \ldots, (2^{n+1} - 1, -2^{n+1} + 1)$. (Bunka $(2^n - 1, -2^{n+1} + 1)$ ožije kvôli tomu, že v jej okolí bude živá bunka $(2^n - 2, -2^{n+1} + 2)$ a nebude živá bunka $(2^n - 1, -2^{n+1} + 2)$, bunka $(2^n, -2^{n+1} + 1)$ ožije, pretože bunka $(2^n, -2^{n+1} + 2)$ bude živá.) □

Dospeli sme k záveru, že pravidlá na oživenie obsahujú číslo 2 a nesmú obsahovať číslo 1. Pravidlá na prežitie živej bunky nemajú žiadne obmedzenia. Dokopy teda máme $(2^6 - 1) \cdot 2^4 = 1008$ pravidiel. Každé z týchto pravidiel sme testovali na desiatich vstupoch, kde sme náhodne vygenerovali počiatočnú konfiguráciu s 50 alebo 100 bunkami na ploche kosoštvorca so stranou 80. Z týchto vstupov sme vypočítali priemernú hodnotu počtu buniek po 30 krokoch a výsledok týchto dvoch testovaní môžeme vidieť na obrázku 5.3.



Obr. 5.3: Graf pre 1008 pravidiel a 100 alebo 50 počiatočných buniek

Kapitola 6

Hľadanie vzorov v šesťuholníkovej mriežke

Tretie Conwayove kritérium hovorí o tom, že existuje konfigurácia, ktorá je nemenná, osciluje alebo uteká preč. Keďže oscilátory a nemenné konfigurácie sa dajú nájsť pomerne ľahko, zamerali sme sa hlavne na hľadanie vesmírnych lodí.

Budeme pracovať s programom umožňujúcim simuláciu bunkových automatov na šesťuholníkovej mriežke popísaným v Kapitole 4.

6.1 Hľadanie vesmírnych lodí

Pri hľadaní vesmírnych lodí na pravidlách, ktoré sme vybrali v predchádzajúcej kapitole, začneme s prehľadávaním všetkých možných konfigurácií, ktoré sa zmestia do šesťuholníka so stranou dĺžky 3. Počet pravidiel, ktoré budeme skúmať, je 1008. Počet všetkých možných konfigurácií je $2^{19} - 1$.

6.1.1 Prvé testovanie

Keďže používame iba pravidlá, pri ktorých je poloha susedov vzhľadom na testovanú bunku nepodstatná, môžeme vylúčiť z počiatočných konfigurácií tie, ktoré sú otočením alebo zrkadlovým obrazom inej skúmanej konfigurácie. (Otočením získame 6 konfigurácií a zrkadlovým obrazom týchto šiestich konfigurácií dostaneme ďalších 6, čo je dokopy nanajvýš 12[1] rovnako sa správajúcich konfigurácií.) Z každej z týchto 12 konfigurácií tes-

[1] Niektoré konfigurácie sú symetrické a tým pádom sú niektoré otočenia alebo zrkadlové obrazy rovnaké.

tujeme práve jednu a to tú, ktorá je kódovaná najmenším číslom. Každú konfiguráciu C kódujeme číslom $\langle C \rangle$ nasledovne. Zakódujeme 19 buniek v šesťuholníku so stranou dĺžky 3 číslami[2] $0, \ldots, 18$ a číslo konfigurácie bude $\langle C \rangle = \sum_{x \in C} 2^{\langle x \rangle}$, kde $\langle x \rangle$ je kód (živej) bunky x. Môžeme si všimnúť že toto číslo má v dvojkovej sústave najviac 19 cifier.

Keďže hľadáme vesmírne lode, rozhodli sme sa ukončiť testovanie konfigurácie ak v niektorom kroku je počet živých buniek 0 (vtedy konfigurácia vymrie) alebo počet živých buniek presiahne číslo 200. (Keďže sme testovali mnoho konfigurácii, testovanie bolo časovo náročné a preto sme zvolili hranicu 200 buniek. Navyše sme chceli nájsť vesmírne lode, pri ktorých je šanca, že budú vygenerované náhodne, preto sme chceli nájsť hlavne vesmírne lode s nižším počtom živých buniek.)

Pre každú konfiguráciu sme odsimulovali 100 krokov na všetkých pravidlách, pričom sme sa pozerali na počty živých buniek v n-tom kroku. Hľadali sme periódu v počte živých buniek, ktorá by napovedala, že počiatočná konfigurácia môže byť vesmírna loď alebo oscilátor.

Výsledkom tohto testovania sú pravidlá, ktoré neskončili z dôvodu vymretia alebo z dôvodu premnoženia a majú periódu počtu živých buniek rôznu od nuly.

6.1.2 Druhé testovanie

Prvým testovaním sme znížili počet testovaných konfigurácií. Keďže chceme nájsť vesmírne lode, t.j. vzory pohybujúce sa po mriežke, môžeme vylúčiť konfigurácie, ktoré majú stály počet buniek 2 alebo 3 na základe vety 6.1.1 a vety 6.1.2.

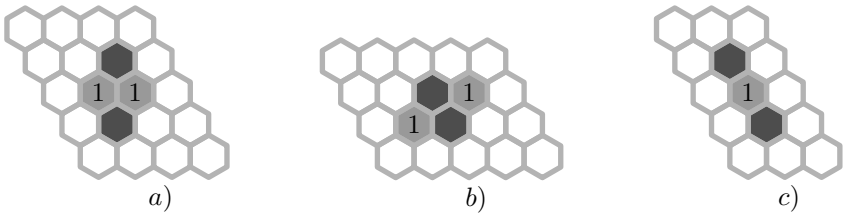
Veta 6.1.1. *Neexistuje vesmírna loď, ktorá má v každom kroku živé práve 2 bunky.*

Dôkaz. Všetky možnosti ako môžeme usporiadať 2 bunky na sieti môžeme vidieť na obrázku 6.1. Dôvodom, prečo nemôžeme umiestniť dve bunky ďalej od seba je to, že by neboli v okolí žiadnej bunky obe naraz. Preto by nemohla ožiť žiadna zatiaľ mŕtva bunka. (Vylúčili sme pravidlá na oživenie s jedným živým susedom.) Keďže by nemohla ožiť žiadna bunka, a dve živé bunky nie sú vo vzájomnom okolí, konfigurácia by po prvom kroku vymrela.

Tmavé bunky na obrázku 6.1 sú živé bunky v prvom kroku. V druhom kroku majú byť tiež živé len dve bunky. Vieme, že každé pravidlo oživí mŕtvu bunku s dvomi živými susedmi. Z toho vyplýva, že šedé bunky s číslom 1 budú v druhom kroku určite živé. Z konfigurácie a) sa po prvom kroku dostaneme do konfigurácie b). (Keďže už vieme o dvoch živých bunkách, žiadna iná bunka nemôže prežiť, pretože by táto vesmírna loď nemala práve

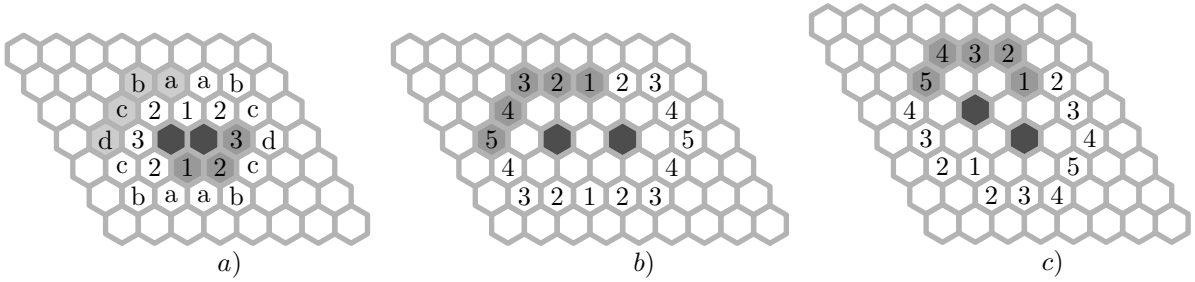
[2] na konkrétnom poradí nezáleží, my sme zvolili číslovanie po riadkoch zhora nadol

2 živé bunky v každom kroku.) Naopak, z konfigurácie b) sa po jednom kroku dostaneme do konfigurácie a). Je preto zrejmé, že táto konfigurácia je oscilátor. V prípade c) po prvom kroku ožije len jedna bunka. Žiadna iná bunka nemôže ožiť, lebo má nanajvýš jedného živého suseda. Ani jedna zo živých buniek nemôže prežiť lebo nemá ani jedného živého suseda. Táto konfigurácia preto po dvoch krokoch vyhynie. □



Obr. 6.1: Počiatočné konfigurácie so stálym počtom buniek 2

Pozrime sa na konfigurácie, ktoré majú tri živé bunky a mohli by byť vesmírne lode. Zrejme nás nemusí zaujímať situácia, keď je niektorá zo živých buniek príliš ďaleko od zvyšných buniek, pretože táto bunka určite zomrie a z vety 6.1.2 vieme, že zo žiadnej dvojprvkovej konfigurácie nedostaneme vesmírnu loď. Pozrime sa preto na všetky možné konfigurácie o troch živých bunkách. Budeme hovoriť, že dve bunky patria do toho istého komponentu, ak sú spojené hranou v grafe susednosti. (Graf susednosti je graf, ktorého vrcholy sú bunky a hrana medzi bunkami a, b je práve vtedy, keď a je v okolí b a naopak, b je v okolí a.) Tri bunky môžu byť v jednom, dvoch alebo troch komponentoch.



Obr. 6.2: Všetky konfigurácie pre 3 bunky

Bunky v jednom komponente môžu byť rozmiestnené tak, ako na obrázku 6.2a). (Dve z buniek sú vyznačené tmavo, tretia bunka je jedna z buniek označených číslami 1, 2, 3.)

V prípade dvoch komponentov sa musia dve bunky dotýkať a tretia bunka musí byť vo vzdialenosti 2 od jednej z týchto buniek. Takýto prípad je znázornený na obrázku 6.2a),

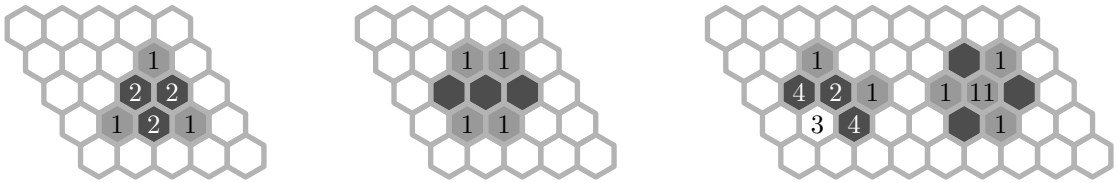
pričom tretia bunka je jedna z buniek označených písmenami a, b, c, d.

Pozrime sa teraz na konfigurácie, ktoré majú tri komponenty. Dve bunky môžu byť oddelené mŕtvou bunkou (obrázok 6.2*b*)) alebo hranou (obrázok 6.2*c*)). Číslami sú označené možné pozície tretej bunky. Môžeme si všimnúť, že prípady *b*)2 a *c*)2, resp. *b*)4 a *c*)4 sú rovnaké.

V nasledujúcom dôkaze sa často vyskytuje prípad, kedy vieme určite povedať, že ožili aspoň tri bunky. Keďže uvažujeme výpočty, v ktorých sú živé práve tri bunky v každom kroku, vieme vyvodiť, že všetky živé bunky zomrú.

Veta 6.1.2. *Neexistuje vesmírna loď, ktorá má v každom kroku živé práve tri bunky.*

Dôkaz. Dôkaz rozdelíme na štyri časti.
a) V prvej časti sa pozrieme na trojice buniek v jednom komponente. Na obrázku 6.3 môžeme vidieť všetky možnosti umiestnenia buniek. V prvom prípade po prvom kroku ožijú tri bunky označené číslom 1. Po druhom kroku ožijú bunky označené číslom 2, tie isté, ktoré boli živé v prvom kroku. Dostávame teda oscilátor. V druhom prípade po prvom kroku majú ožiť až 4 bunky, čo nevyhovuje našim podmienkam a teda z tohto vzoru nevznikne oscilátor so stálym počtom buniek 3. V treťom prípade ožijú 2 bunky s číslom 1. Tretiu bunku nevieme s istotou identifikovať. Dve bunky s číslami 4 nemôžu prežiť, lebo by boli živé až 4 bunky. Ak by prežila bunka s číslom 2, tak po prvom kroku dostaneme počiatočnú konfiguráciu len otočenú o 180 stupňov a teda dostávame oscilátor. Ak by však ožila bunka s číslom 3 vieme, že v pravidlách na oživenie bunky máme pravidlo s číslom 3. Po druhom kroku by však ožili až 4 bunky (bunka s číslom 11 ožije kvôli pravidlu s číslom 3) a teda ani táto konfigurácia nie je vyhovujúca.

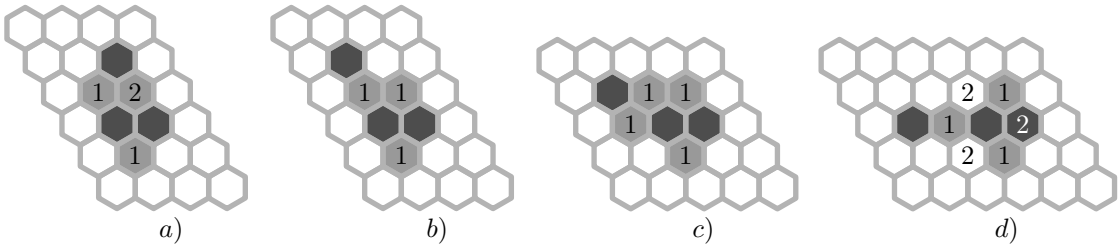


a) *b*) *c*)

Obr. 6.3: Počiatočné konfigurácie so stálym počtom buniek 3, v jednom komponente

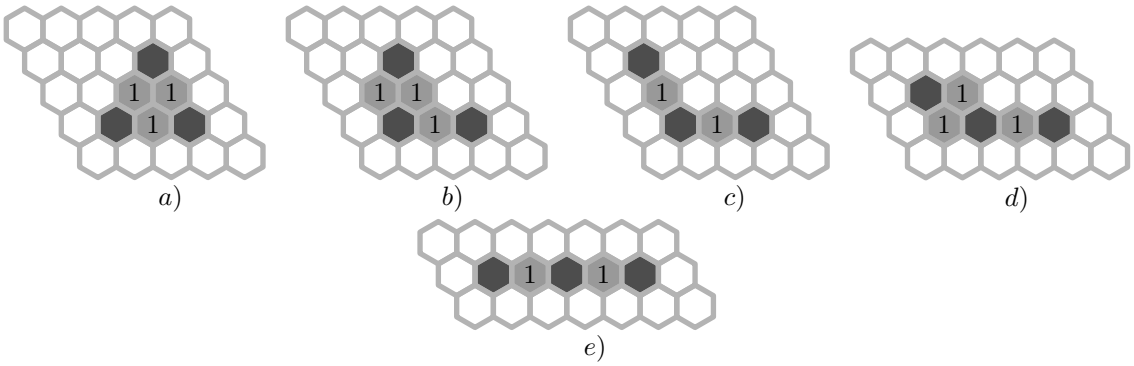
b) V druhej časti sa pozrieme na tri bunky v dvoch komponentoch, ktoré sú nakreslené na obrázku 6.4. V možnosti *a*) po prvom kroku ožijú 2 bunky s číslom 1. Žiadna iná bunka, okrem bunky s číslom 2, nemá živých aspoň dvoch susedov a preto musí táto bunka ožiť. Do pravidiel na oživenie pribudne pravidlo s číslom 3. Po druhom kroku vznikne

konfigurácia, ktorá je otočením počiatočnej konfigurácie o 180 stupňov. Z toho vyplýva, že z tejto možnosti dostávame oscilátor. V možnosti *b*) po prvom kroku dostaneme rovnakú konfiguráciu ako v možnosti *a*). Z tejto možnosti teda vznikne rovnaký oscilátor ako z prvej. V možnosti *c*) po prvom kroku ožijú až 4 bunky, takže táto možnosť nie je dobrá. V možnosti *d*) po prvom kroku ožijú 3 bunky s číslom 1. Po druhom kroku ožijú bunky s číslom 2. Táto konfigurácia je otočením predošlej konfigurácie o 60 stupňov, a teda dostaneme oscilátor.



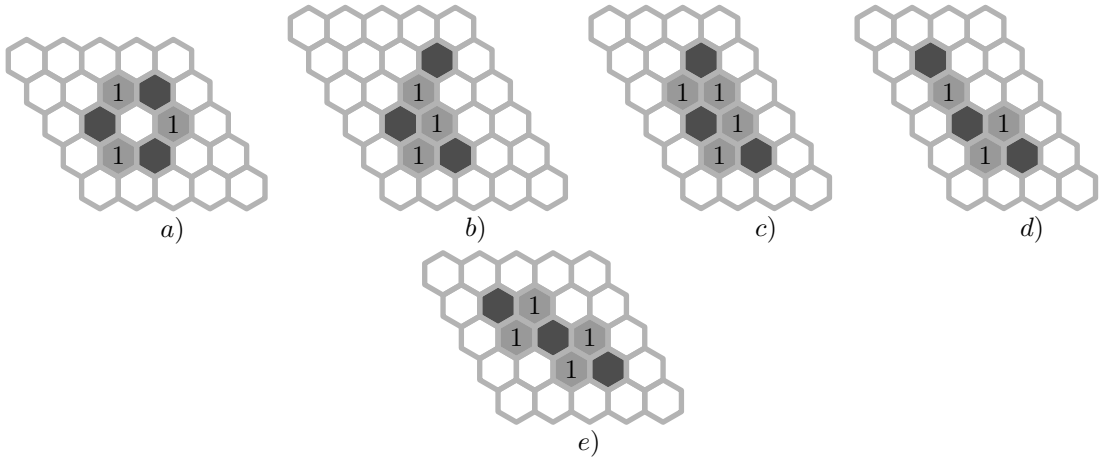
Obr. 6.4: Počiatočné konfigurácie so stálym počtom buniek 3, v dvoch komponentoch

c) V tretej časti sa pozrieme na tri komponenty, kde sú dve bunky oddelené mŕtvou bunkou. (Viď obrázok 6.5) V možnosti *a*) po prvom kroku ožijú 3 bunky, ktoré vytvoria konfiguráciu, ktorú sme už analyzovali a je na obrázku 6.3*a*). Stane sa z nej oscilátor. Z možnosti *b*) sa po prvom kroku stane konfigurácia ako na obrázku 6.3*c*). Z tejto možnosti sa buď stane oscilátor alebo vyhynie. V konfigurácii *c*) ožijú len dve bunky. Žiadna živá bunka nemá aspoň jedného živého suseda aby prežila. Žiadna mŕtva bunka nemá aspoň dvoch živých susedov aby mohla ožiť. Takže táto konfigurácia vyhynie. Z konfigurácie *d*) sa po prvom kroku stane konfigurácia ako na obrázku 6.4*a*). Z tejto konfigurácie vznikne oscilátor. V konfigurácii *e*) zase ožijú len dve bunky a žiadna iná nemá ako ožiť alebo prežiť, a preto táto konfigurácia vyhynie.



Obr. 6.5: Počiatočné konfigurácie so stálym počtom buniek 3, v troch komponentoch

d) V poslednej časti sa pozrieme na tri komponenty, nakreslené na obrázku 6.6, kde sú dve bunky oddelené hranou. V možnosti *a*) po prvom kroku ožijú tri bunky. Táto konfigurácia je rovnaká ako počiatočná, len otočená o 60 stupňov. Z tejto konfigurácie teda vznikne oscilátor. Z možnosti *b*) po prvom kroku dostaneme možnosť ako na obrázku 6.3*c*). Vznikne z nej oscilátor alebo vymrie. V možnosti *c*) po prvom kroku ožijú až 4 bunky, táto konfigurácia nie je dobrá. Z konfigurácie *d*) po prvom kroku dostaneme konfiguráciu ako na obrázku 6.4*b*). Vznikne z nej oscilátor. V konfigurácii *e*) ožijú po prvom kroku 4 bunky takže z nej tiež nevznikne vesmírna loď s konštantným počtom buniek. □



Obr. 6.6: Počiatočné konfigurácie so stálym počtom buniek 3, v troch komponentoch

Ukázali sme, že vieme niektoré konfigurácie vylúčiť na základe prvého testovania a viet 6.1.1 a 6.1.2. Zvyšné konfigurácie otestujeme podobne ako v prvom testovaní, len budeme simulovať 300 krokov výpočtu. Týmto testom chceme zreteľnejšie odlíšiť oscilátory od vesmírnych lodí.

Pomocou tohto testovania sme vylúčili niekoľko ďalších konfigurácií, ktoré vymreli alebo mali počet živých buniek väčší ako 200. Navyše sme čiastočne oddelili vesmírne lode od oscilátorov tým, že vesmírne lode budú po 300 krokoch výpočtu ďalej od bodu $(0, 0)$ ako oscilátory.

6.2 Výsledky testovaní

Počas testovania sa nám podarilo objaviť 17 nových vesmírnych lodí, ktoré môžeme vidieť v prílohe A. Pri každom obrázku sú napísané pravidlá, pre ktoré sa konfigurácia správa

rovnako a v zložených zátvorkách je uvedená perióda tejto vesmírnej lode.

Na obrázku A.1*a*) môžeme vidieť čo do rozlohy ale i počtu buniek najväčšiu vesmírnu loď, ktorú sa nám podarilo nájsť. Na tomto obrázku môžeme tiež vidieť ďalších 7 vesmírnych lodí, ktorých konfigurácie v iných krokoch sme neuviedli vzhľadom na veľkosť ich periódy. Na obrázkoch A.2, A.3 a A.4 však môžeme vidieť 9 vesmírnych lodí v každom kroku ich periódy a tiež posun konfigurácie počas jednej periódy.

Vieme, že Game of Life na náhodných konfiguráciách obsahuje viacero vesmírnych lodí. Ak chceme aby sa vesmírne lode mali šancu vygenerovať náhodne, musia byť malé. (Mať malý počet živých buniek v konfigurácii.) Túto podmienku najviac spĺňajú nasledovné pravidlá: 245/2456 a 2456/2456, ktoré podporujú vesmírne lode na obrázku A.3*a*), A.2*c*) a A.4*a*). Pravidlo 2456/2456 navyše podporuje vesmírne lode na obrázkoch A.2*b*) a A.4*b*). Ďalšie pravidlá, ktoré fungujú na dvoch vesmírnych lodiach sú 26/2 (A.1*a*), A.2*a*)), 256/25 (A.1*c*), A.2*a*)), 2/26 (A.1*b*), A.2*a*)) a všetky nasledovné pravidlá: 2456/246, 2456/245, 2456/24, 245/246, 245/245, 245/24 fungujú pri vesmírnych lodiach (A.3*a*), A.2*c*)).

Kapitola 7

Záver

V práci sme sa zaoberali hľadaním ekvivalentných pravidiel ku hre Game of Life, ktoré by fungovali na šesťuholníkovej mriežke. Náš prístup zahŕňal splnenie Conwayových podmienok. Používali sme okolie šiestich buniek, dvojstavový automat a triedu pravidiel, ktorá sa rozhoduje len podľa počtu živých buniek v okolí.

Pomocou prvých dvoch podmienok sme vylúčili značné množstvo pravidiel, ktoré nemôžu byť považované za ekvivalent ku hre Game of Life. Pri snahe vyhovieť tretej Conwayovej podmienke sme sa zamerali na hľadanie vesmírnych lodí, ktorých hľadanie je náročnejšie ako hľadanie oscilátorov, či nehybných vzorov.

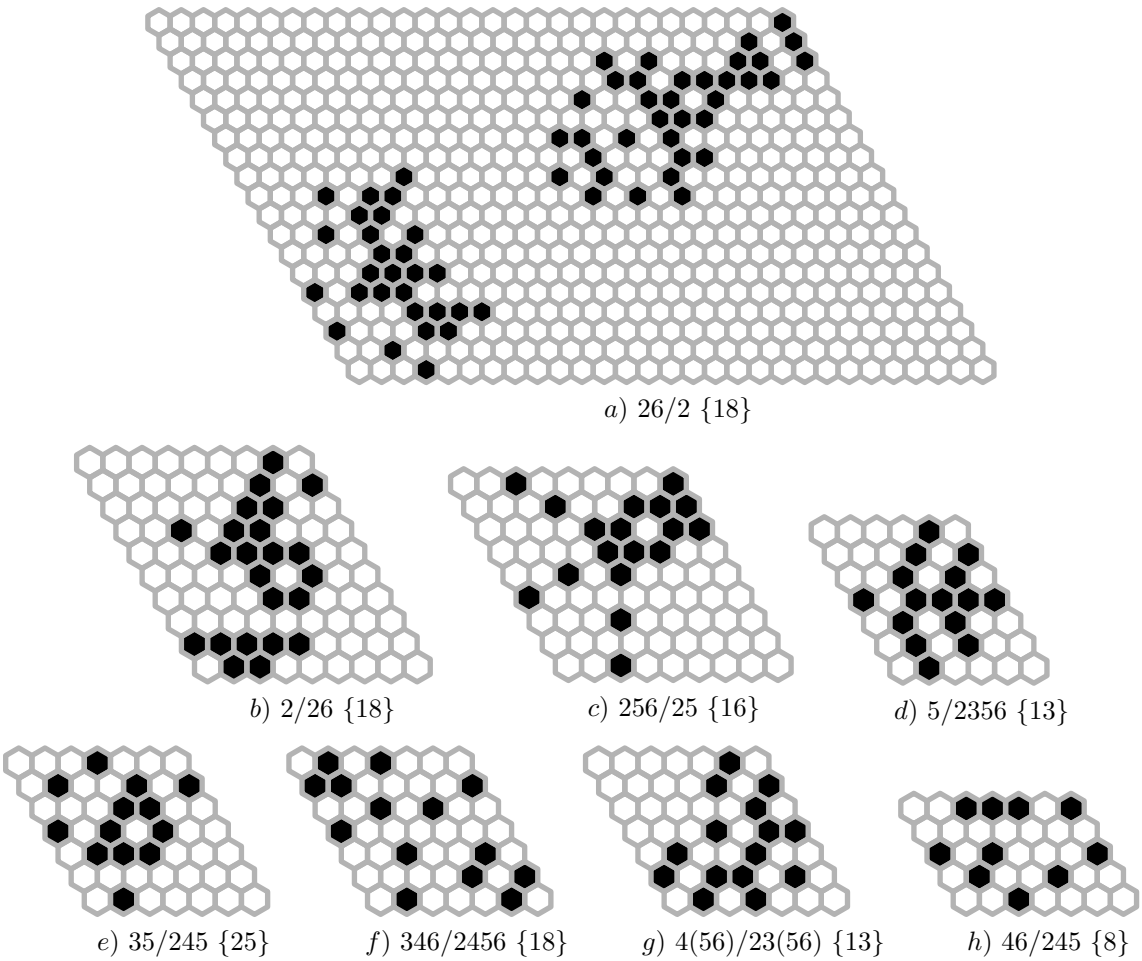
V rámci práce sa nám podarilo nájsť 17 vesmírnych lodí, ktoré sme nenašli v inej literatúre. Jedna z vesmírnych lodí má dokonca najviac 58 živých buniek a periódu 18. Vesmírna loď s najväčšou periódou, ktorú sme našli, má periódu 25. Nie je ľahké vybrať vhodný ekvivalent ku Game of Life len podľa počtu vesmírnych lodí. Pravidlá, ktoré podporujú najviac vesmírnych lodí sú pravidlá 245/2456 a 2456/2456. Ak sa však pozrieme na pôvodné Conwayove pravidlá a na ich biologickú motiváciu, tieto pravidlá sa nám nezdajú najvhodnejšie. Preto nie je správne domnievať sa, že na základe našich výsledkov sú tieto pravidlá najlepšie alebo jediné možné ekvivalenty Game of Life na šesťuholníkovej mriežke.

Hľadanie ekvivalentných pravidiel môže byť vykonávané aj z iného pohľadu akým sme sa riadili my. Pre Conwayovu Game of Life napríklad platí, že náhodné konfigurácie „vybuchujú“. Iným prístupom môžeme napríklad porovnávať štatistické výsledky pravidiel na štvorčekových a šesťuholníkových mriežkach a vybrať to šesťuholníkové pravidlo, ktoré bude najviac podobné pravidlu pre Game of Life.

Okrem iných pohľadov uvedených vyššie, je možné zaoberať sa aj zložitejšími pravidlami (pravidlá závislé od času, pravidlá, ktoré uvažujú aj pozíciu živých buniek, a pod.), iným okolím, prípadne automatmi s viacerými stavmi. Takéto prístupy sú bežné v iných prácach, my si však myslíme, že je vhodné najprv preskúmať jednoduchý model, aby sa zachovala podobnosť s Conwayovou Game of Life.

Dodatok A

Vesmírne lode na šesťuholníkovej mriežke



a) 26/2 {18}

b) 2/26 {18}

c) 256/25 {16}

d) 5/2356 {13}

e) 35/245 {25}

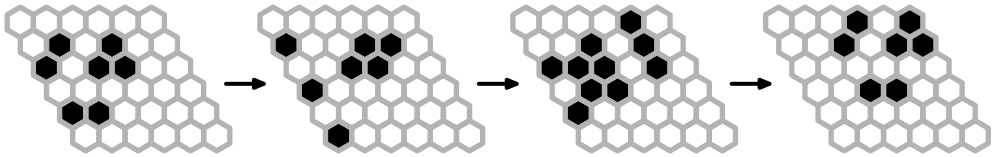
f) 346/2456 {18}

g) 4(56)/23(56) {13}

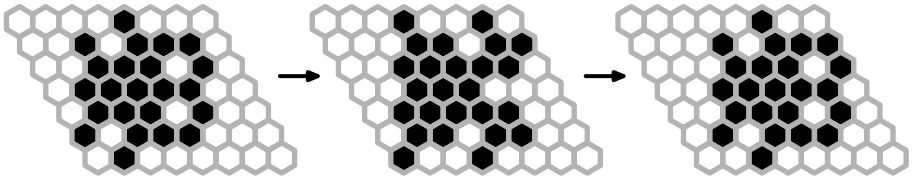
h) 46/245 {8}

Obr. A.1: Vesmírne lode 1

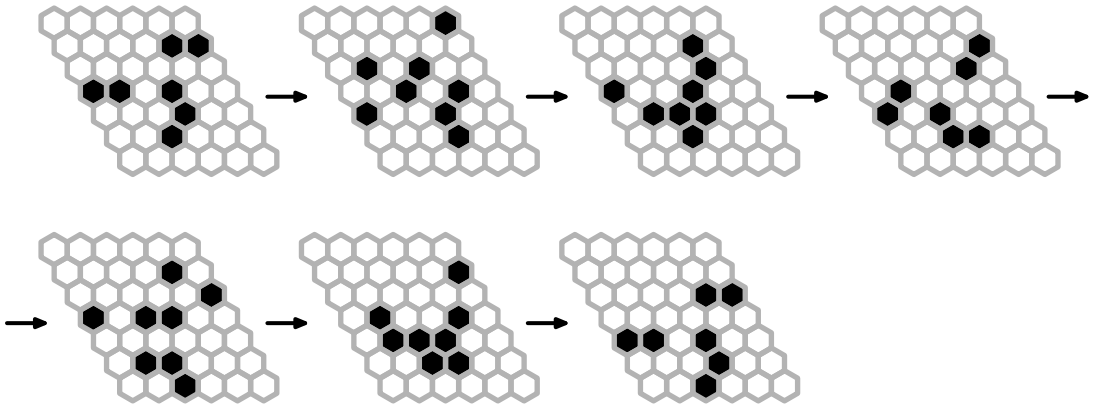
a) Pravidlo 2(56)/2(56) {3}



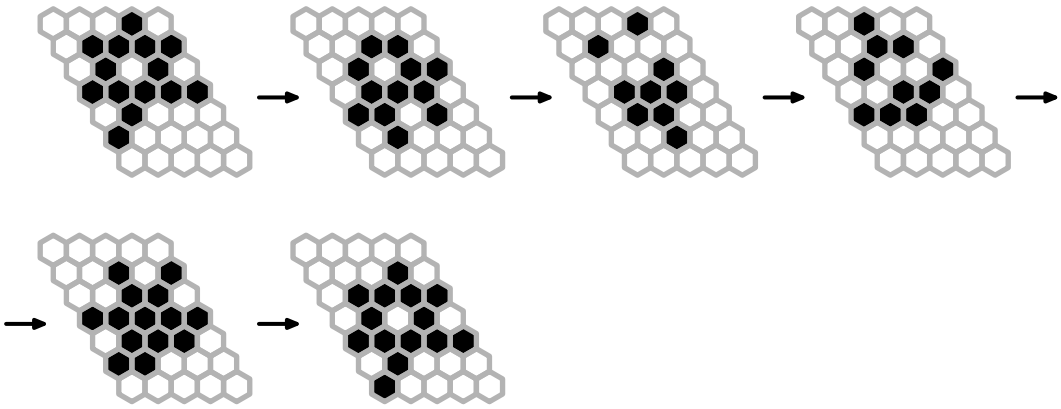
b) Pravidlo 2456/2456 {2}



c) Pravidlo 24(56)/24(56) {6}

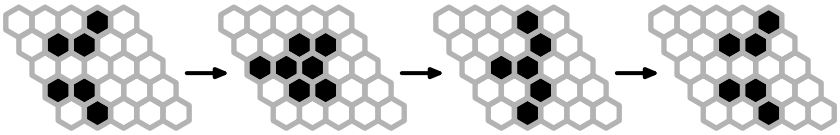


d) Pravidlo 345/245 {5}

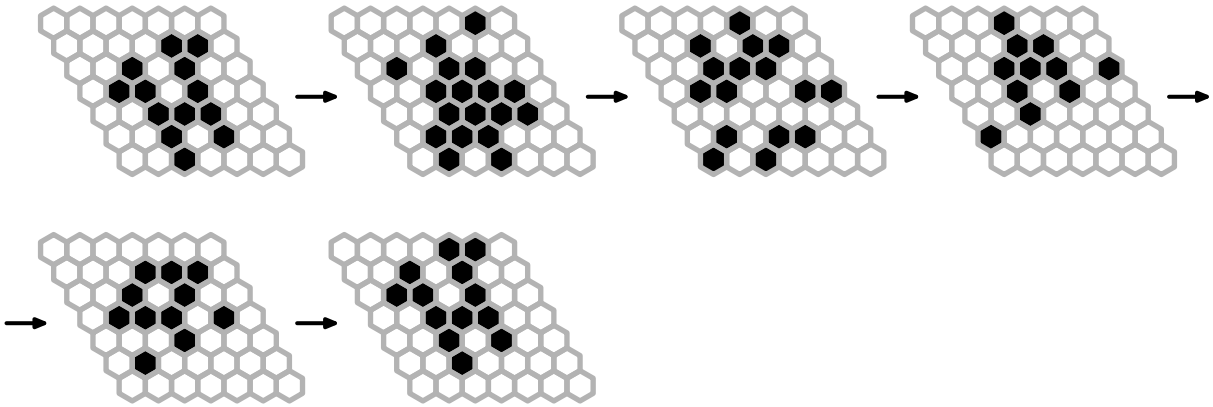


Obr. A.2: Vesmírne lode 2

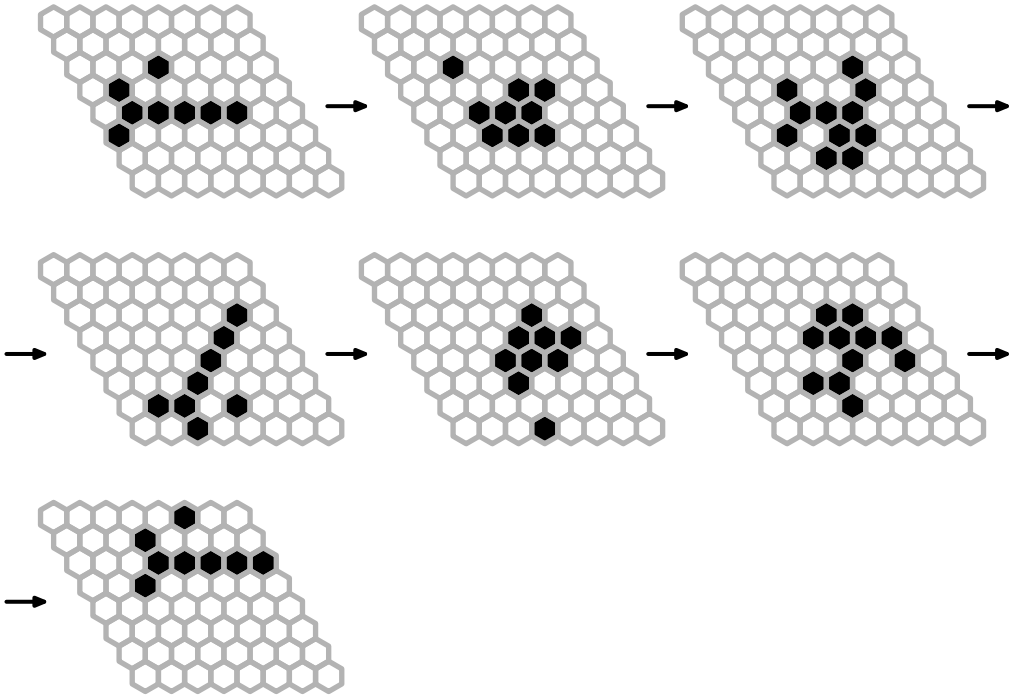
a) Pravidlo 245(6)/24(56) {3}



b) Pravidlo 34/245 {5}

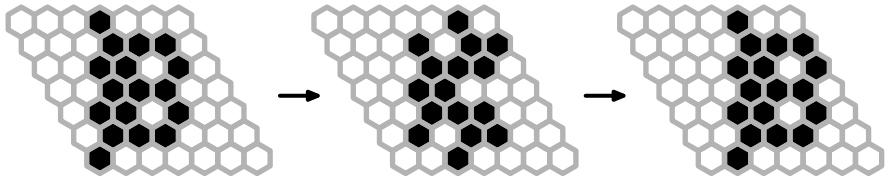


c) Pravidlo 245(6)/2(6) {6}

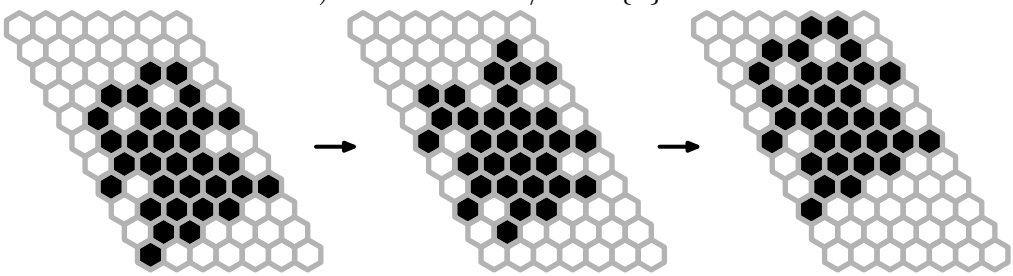


Obr. A.3: Vesmírne lode 3

a) Pravidlo 245(6)/245(6) {2}



b) Pravidlo 2456/2456 {2}



Obr. A.4: Vesmírne lode 4

Literatúra

[Atr65] A. J. Atrubin. A One-dimensional Real-Time Iterative Multiplier. *IEEE Trans. Computers*, pages 14–394, 1965.

[Bay02] Carter Bays. Cellular Automata and the Game of Life in the Hexagonal Grid. `http://www.cse.sc.edu/~bays/h6h6h6/`, 2002.

[BBK98] H. Baltzer, W. P. Braun, and W. Kohler. Cellular Automata Model for Vegetable Dynamic. *Ecological Modelling*, 107:113–125, 1998.

[BF84] C. Burks and D. Farmer. Towards modeling DNA sequences as automata. *Physica D Nonlinear Phenomena*, 10:157–167, jan 1984.

[CCF93] B. Chopard, Alfonso Caiazzo, and Jean-Luc Falcone. Cellular automata approach reaction-diffusion system. *Cellular Automata Prospects in Astrophysical Applications*, pages 157–186, 1993.

[CDK89] B. Chopard, M. Droz, and M. Kolb. Cellular automata approach to non-equilibrium diffusion and gradient percolation. *Journal of Physics A Mathematical General*, 22:1609–1619, may 1989.

[Col69] S. N. Cole. Real-Time Computation by n-Dimensional Iterative Arrays of Finite-State Machines. *IEEE Trans. Computers*, 1969.

[FHP86] U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-Gas Automata for the Navier-Stokes Equation. *Phys. Rev. Lett.*, 56(14):1505–1508, Apr 1986.

[Fis65] Patrick C. Fischer. Generation of Primes by a One-Dimensional Real-Time Iterative Array. *J. ACM*, 12(3):388–394, 1965.

[GM02] D. Griffeath and C. Moore(ed). New Constructions in Cellular Automata. 2002.

[Goo05] Google. Google SparseHash template. `http://code.google.com/p/google-sparsehash/`, 2005.

[GSD+03] Niloy Ganguly, Biplab K Sikdar, Andreas Deutsch, Geoffrey Canright, and P Pal Chaudhuri. A Survey on Cellular Automata. Technical report, Centre for High Performance Computing, dec 2003.

[GW95] Richard J. Gaylord and Paul R. Wellin. *Computer simulations with Mathematica: explorations in complex physical and biological systems.* Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[Kar05] Jarkko Kari. Theory of cellular automata: A survey. *Theoretical Computer Science*, (1-3):3–33, April 2005.

[KO93] Donald C. Keenan and Mike J. O'Brien. Competition, collusion, and chaos. *Journal of Economic Dynamics and Control*, 17:327–353, May 1993.

[lBoHMD98] Jean luc Beuchat, Jacques olivier Haenni, Student Member, and John Von Neumann Designed. Von Neumann's 29-state cellular automaton: A hardware implementation. Manuscript submitted for publication, 1998.

[LG90] Fong Liu and Nigel Goldenfeld. Generic features of late-stage crystal growth. *Phys. Rev. A*, 42:895–903, Jul 1990.

[Man77] F. B. Manning. An Approach to Highly Integrated, Computer-Maintained Cellular Arrays. *IEEE Trans. on Computers*, C-26:536–552, 1977.

[May03] George Maydwell. Hexagonal cellular automata techniques. `http://www.collidoscope.com/ca/hcat.html`, 2003.

[MF83] Barry F. Madore and Wendy L. Freedman. Computer simulations of the Belousov-Zhabotinsky reaction. *Science*, 222:615–616, 1983.

[Neu63] J. V. Neumann. *Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components.* 1963.

[Nis81] H. Nishio. Real Time Sorting of Binary Numbers by One-dimensional Cellular Automata. 1981.

[NK75] H. Nishio and Y. Kobuchi. Fault Tolerant Cellular Space. *J. Comput. Syst. Science*, 11:150–170, 1975.

[Pac86] N. H. Packard. Lattice Models for Solidification and Aggregation. 1986.

[PDL+79] Jr. Preston, K., M.J.B. Duff, S. Levialdi, P.E. Norgren, and J. Toriwaki. Basics of cellular logic with some applications in medical image processing. *Proceedings of the IEEE*, 67:826–856, May 1979.

[Res94] Mitchel Resnick. *Turtles, termites, and traffic jams.* MIT Press, Cambridge, MA, USA, 1994.

[Ros79] Azriel Rosenfeld. *Picture Languages.* Academic Press, Inc., 1979.

[RU02] G. A. Raboni and A. Laghi. Urban. Cellular Automata: The Inverse Problem. pages 345–356, October 2002.

[Sar00] Palash Sarkar. A brief history of cellular automata. *ACM Comput. Surv.*, pages 80–107, 2000.

[Sch69] Thomas C. Schelling. Models of Segregation. *American Economic Review*, 59(2):488–493, May 1969.

[Sch71] Thomas C. Schelling. Dynamic models of segregation. *Journal of Mathematical Sociology*, 1(2):143–186, June 1971.

[SGI97] SGI. Standard Template Library Programmer's Guide. `http://www.sgi.com/tech/stl/set.html`, 1997.

[SK02] Birgitt Schönfisch and Michael Kinder. A Fish Migration Model. Lecture Notes in Computer Science, pages 210–219. Springer, October 2002.

[S.R80] S.R.Sternberg. Language and Architecture for Parallel Image Processing. *Physica D Nonlinear Phenomena*, page 35, 1980.

[SWH84] S. A. Smith, R. C. Watt, and S. R. Hameroff. Cellular automata in cytoskeletal lattices. *Physica D Nonlinear Phenomena*, 10:168–174, jan 1984.

[Weia] Eric W. Weisstein. Moore Neighborhood. `http://mathworld.wolfram.com/MooreNeighborhood.html`. From MathWorld – A Wolfram Web Resource.

[Weib] Eric W. Weisstein. von Neumann Neighborhood. `http://mathworld.wolfram.com/vonNeumannNeighborhood.html`. From MathWorld – A Wolfram Web Resource.

[Wik05] WikiBooks.org. Cellular Automata/Mathematical Model. `http://en.wikibooks.org/wiki/Cellular_Automata/Mathematical_Model`, 2005.

[Wol59] Stephen Wolfram. *A New Kind of Science.* Wolfram Media Inc., 1959.

[WWS85] A. T. Winfree, E. M. Winfree, and H. Seifert. Organizing centers in a cellular excitable medium. *Physica D Nonlinear Phenomena*, 17:109–115, aug 1985.

[You84] D. Young. A Local Activator-Inhibitor Model of Verterbate Skin Patterns. *Math. Biosciences*, 72:51–58, 1984.

[ZZ96] Q. Zeng and X. Zeng. An Analytical Dynamic Model for Grass Field Ecosystem. *Ecological Modelling*, 85:187–196, 1996.