Samoorganizace a Buněčné Automaty

Martin Kočica

Bakalářská práce
2005

Univerzita Tomáše Bati ve Zlíně
Fakulta technologická

ABSTRAKT

Tato práce skýtá krom rešerše v oblasti buněčných automatů zaměřené na samoorganizaci také přehled různých druhů buněčných automatů a jejich praktické využití. Její podstatnou částí je též rozbor známého automatu „Život“ a útvarů, které v něm vytváří samoorganizace, i těch, které byly nalezeny až studiem tohoto fenoménu. Praktická část sestává z programů v prostředí Mathematica, které umožňují realizaci semi-totalistického automatu s jakýmikoliv pravidly ve dvou a třech dimenzích.

ABSTRACT

In addition to the recherche in the field of cellular automata and self-organization this project contains a brief survey of one- and two-dimensional CA and examples of their application. Much attention is also paid to the famous CA „Game of Life“ and patterns whic either appear in its environment as a product of self-organization or were created by numerous researchers. The practical part consists of two programs which enable simulation of any semi-totalistic two- or three-dimensional CA in Mathematica.

OBSAH

ÚVOD

Svět kolem nás se skládá z množství nejrůznějších částí, které bez zásahu člověka fungují jako dokonalý systém. Jednotlivé prvky jsou ve vzájemném souladu a přesně navazují jeden na druhý. Výklady o původu tohoto systému jsou od pradávna objektem zájmu nejrůznějších myslitelů, filozofů, náboženských vůdců či moderních vědců. Je opředen řadou mýtů a tajemství jež se po malých krůčcích snaží lidstvo rozluštit. Mnohé teorie si zakládají na existenci vyšší síly, která veškeré součásti skloubila dohromady a stanovila pro celý systém dokonalý řád. V historii to ale není jediný pohled na problém. S myšlenkou, že uspořádanost systému může být způsobena samotnou dynamikou systému si pohrávali filozofové již ve starém Řecku. Idea prošla v průběhu času množstvím obměn a postupně vedla až k pojmu samoorganizace. Jak se ukazuje, může řád v systému vznikat bez vnějšího působení pouhou interakcí jednotlivých jeho prvků, což lze snadno vypozorovat například u buněčných automatů.

Buněčný automat je matematický model sloužící k simulaci především nejrůznějších přírodních dějů. Procesy v něm probíhající představují typický příklad samoorganizace, neboť zcela chaotická počáteční konfigurace se během procesu transformuje na organizované útvary. Mnohé z těchto útvarů lze dále využít a vytvořit jiný systém schopný realizovat rozličné výpočty.

Rozmanitost buněčných automatů a útvarů které v nich lze nalézt umožňuje jejich další využití v univerzálních výpočtech algoritmicky zapsatelných problémů. Proto je vhodné zabývat se nejen studiem buněčných automatů jako systému, ale i v něm vznikajících struktur.

I. TEORETICKÁ ČÁST

1 HISTORIE

1.1 Samoorganizace

Myšlenka, že dynamika systému může inklinovat k vnitřní uspořádanosti (systému) má dlouhou historii. Již starořečtí atomisté tvrdili, že při dostatku času, prostoru a hmoty je organizace nakonec nevyhnutelná, ačkoli ji v systému nic nevynucuje. Později filozof Descartes ve svém díle formuloval tvrzení, že běžné přírodní zákony směřují k organizovanosti systémů.

Počátkem osmnáctého století vzrůstala snaha o pochopení „univerzálních zákonů chování“, přesněji o vysvětlení vypozorovaných pravidel života organismů. Spojitost myšlenek s ideami Lamarckismu však toto téma na dlouhou dobu stavěla mimo uznávanou vědeckou problematiku. Až počátkem dvacátého století, zásluhou průkopníků jako D’Arcy Wentworth Thomson, došlo ke znovuoživení problematiky. V moderním pojetí věci se vskutku připouští, že existují jakási univerzální pravidla která řídí růst a podobu u biologických systémů.

Pojem „samoorganizace“ byl pravděpodobně prvně prezentován v roce 1947 psychiatrem a inženýrem W. Rossem Ashbym. Samoorganizace jako pojem a koncept byl použit v asociaci s obecnou teorií systémů v šedesátých letech dvacátého století. Skutečně užívaným se pojem stal ale až po převzetí fyziky a výzkumníků na poli komplexních systémů v sedmdesátých a osmdesátých letech.[5]

1.2 Buněčné automaty

Historie buněčných automatů sahá do počátků dvacátého století. Turingův stroj, objevený v roce 1936 byl založen na myšlence libovolných operací nad sekvencí diskrétních elementů.[5] Nástup rozličných elektronických počítačů v padesátých letech dvacátého století vedl k sestavení množství systémů podobných buněčným automatů. Ovšem samotná myšlenka, zda stroj může vytvořit kopie sama sebe, se rozrostla až později a to především díky nástupu kybernetiky.

Do širšího podvědomí vstoupily buněčné automaty v roce 1940, kdy byly představeny Stanislavem Ulmanem. Na jeho dílo navázal jeho spolupracovník John von Neumann,

který se ve své práci pokoušel nalézt abstraktní model samoreprodukce u biologických systémů. Přestože počáteční úvahy rozvíjel von Neumann na 3-dimenzionálních modelech vyjádřených parciálními diferenciálními rovnicemi, za nedlouho dospěl k závěru, že 2-dimenzionální prostor bude pro jeho myšlenky dostačující. Inspirován návrhy Stanislawa Ulmana z roku 1951, zjednodušil sestavený model a vypracoval 2-dimenzionální buněčný automat. Vzhledem k nemožnosti fyzicky sestavit sebe-replikující stroj, využil možnosti implementovat tento stroj jako algoritmus v podobě právě 2D buněčného automatu. Výsledkem byl univerzální systém (UCC – Universal Copier and Constructor) pracující s malým okolím sousedů a 29 stavy pro každou buňku. Matematicky dokázal, že mnohé předlohy vytváří nekonečný počet kopií sebe sama uvnitř existujícího vytyčeného prostředí.[1]

Bezprostředně se von Neumannova práce objevuje jako základ pro diskusi ve dvojici odlišných námětů. První, časově zařaditelný na počátek šedesátých let dvacátého století, se týká sestrojení skutečného sebe-replikujícího se stroje, nejčastěji v podobě vesmírného plavidla. Druhý se zaměřil na matematickou podstatu buněčných automatů.

Díky těmto snahám se podařilo v průběhu šedesátých let vypracovat modely automatů s mnohem jednodušší strukturou schopných sebe-reprodukce a univerzálních výpočtů. Příkladem takto zjednodušených konstrukcí je simulace, kterou na počítači PDP-1 provedl Eduard Fredkin kolem roku 1961. Při své práci zaznamenal sebe-reprodukující vlastnosti na modelu analogickém pravidlu 90 v 2D prostoru.

V roce 1970 se stal díky Martinu Gardnerovi velmi populárním dvoustavový, dvojdimenzionální buněčný automat, jež dostal název „Game of Life“, který poprvé popsal John Conway. Při jednoduchosti svých pravidel dává množství rozličných variant následníků. Conway se k výslednému automatu propracoval po sérii pokusů na různých 2D buněčných automatech s rozličnými pravidly.

V roce 1969 Konrad Zuse publikoval ve své knize myšlenku, že fyzikální zákony světa nejsou spojité, ale že veškerý svět je pouze výstup deterministického výpočtu gigantického automatu. Zaměřil se na buněčné automaty jako na možný model. Poukázal na skutečnost, že klasické pojetí růstu entropie nedává smysl u deterministicky odvozeného světa. Jinými

slovy, kde není vůbec přítomná náhoda, nelze mluvit o růstu entropie. Viz buněčný automat jako příklad deterministického universa.[5][1]

Stephen Wolfram prezentoval v roce 1983 sérii publikací v nichž systematicky prozkoumal základní třídy buněčných automatů, jež nazval elementárními buněčnými automaty.

2 CHARAKTERISTIKA A KONSTRUKCE BUNĚČNÝCH AUTOMATŮ

Buněčný automat je diskrétní, paralelně pracující matematický model struktur z oblasti především biologie či chemie. Zejména díky vývoji výpočetní techniky rostou i možnosti simulace těchto modelů a narůstají tak i možnosti využití. [5]

Buněčný automat je složen z nekonečné mřížky buněk, z nichž každá má konečný počet stavů. Mřížku lze definovat v jakékoli konečné dimenzi. S rostoucím počtem dimenzí roste i bohatost automatů, ale také jejich složitost. Čas je definován v diskrétních okamžicích, přičemž stav buňky v daném čase je funkcí stavu konečného počtu buněk, nazývaných sousedy, v čase bezprostředně předcházejícím. Funkci jež definuje stav buňky nazýváme pravidly. Tato pravidla jsou jednotně definována pro všechny buňky a jejich nosným základem je aktuální stav sousedů.

Provede-li se aktualizace celé mřížky buněk vzniká nová generace, jež je základem pro další, iterativně se navyšující počet generací.

S pravidly a generací úzce souvisí i pojem reversibility. V okamžiku, kdy dojde k vytvoření nové generace buněk, je ta stará, není-li externě uchovávána, v samotném systému zapomenuta. V případě, že pravidla zaručují vždy možnost jednoznačného určení předchozí generace, jedná se o reversibilní buněčný automat. Většina automatů však tuto vlastnost nemá a jedná se o automaty ireversibilní. Jarkko Kari a další badatelé postupně dokázali, že pouze u jednodimenzionálního buněčného automatu lze rozhodnout o jeho reversibilitě, u dvou a více dimenzionálních toto rozhodnutí provést nelze. Tato vlastnost je klíčová například při využití buněčných automatů v kryptografii.[5]

Je obvyklé, že v počáteční generaci jen nepatrné procento buněk má jiný stav než zbytek mřížky. Toto rozmístění je startovací konfigurací automatu a má významný vliv na celý jeho následující běh.

Ač není buněčný automat nikterak prostorově krácen, při simulaci je časté omezení na konečnou mřížku. U dvou-dimenzionálního systému se proto spíše než s nekonečnou plochou kalkuluje s obdélníkovou výsečí. Kraje tohoto obdélníka se spojují v toroid, čímž vzniká alternativa nekonečného prostoru, neboť za sousedy lze v takto vytvořeném

prostoru považovat buňky ležící na horním a spodním okraji tohoto obdélníka, případně na jeho pravé a levé straně.

Jinou variantou, se kterou se lze setkat při ošetření okrajů omezeného prostoru, je definování virtuálního rozšíření okrajů o jednu řadu buněk, jejichž hodnota neustále setrvává na nule. Tyto buňky jsou zahrnuty do výpočtu nové generace pouze jako sousedé krajních buněk automatu, pro ně samotné se žádný výpočet neprovádí. U ostatních dimenzí je konstrukce v omezeném prostoru obdobná.

Se systémy buněčných automatů je úzce spjat pojem samoorganizace. Tento proces vnitřní organizace systému bez vnějšího zásahu či externí předlohy lze přímo chápat jako jejich součást.[1][3][4][5]

2.1 Jednodimenzionální buněčné automaty

2.1.1 Elementární buněčné automaty

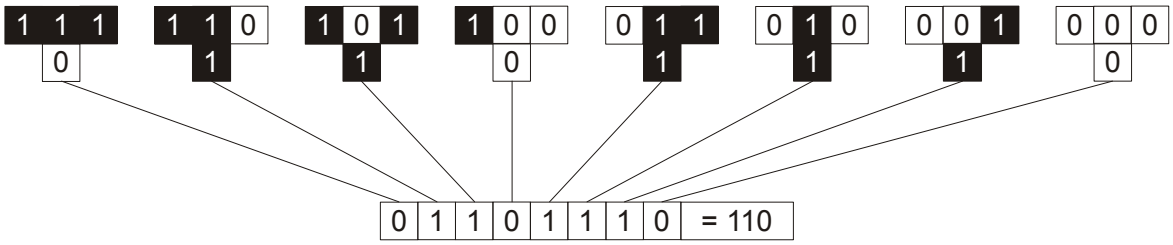
Nejjednodušší netriviální jednorozměrné modely, s nimiž se lze setkat, tvoří takzvané elementární buněčné automaty. Základním kamenem těchto automatů je dvoustavová buňka s dvojicí svých sousedů. Za sousedy se zde považují buňky bezprostředně buňku obklopující.

Obr. 1. Wolframovská sousednost

Tato trojice, vzhledem k počtu možných stavů, umožňuje vytvoření 2^3 různých seskupení. Na základě těchto variací lze sestavit 2^8 odlišných pravidel pro generování nové, 1-dimenzionální generace. Pravidla byla klasifikována S. Wolframem, který také navrhl jejich názvy. Pojmenování vychází z formy zápisu a hodnoty, kterou udává binární záznam samotného pravidla.[2]

Každé pravidlo elementárního buněčného automatu lze zapsat do podoby tabulky o dvou řádcích a osmi sloupcích. Horní řádek, jehož podoba je pro všechna pravidla konstantní, specifikuje aktuální stav buňky a jejích sousedů, zatímco spodní, stav buňky v nové

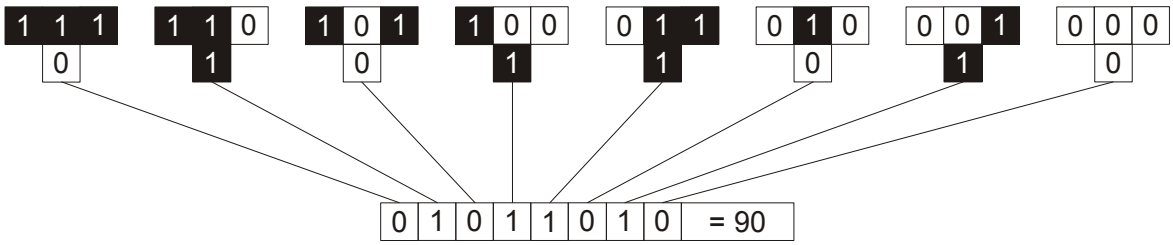
generaci. Z obrázku je již patrné, jak dostalo své jméno například pravidlo 110, Wolframem označováno jako „rule 110 CA“. pravidlo 110.[4]



Obr. 2. Pravidlo 110

Pravidla jednoznačně charakterizují novou generaci buněk, definujíce ji na základě předchozího stavu. Při vytváření další generace dochází k prostému porovnávání aktuálního stavu buňky a jejích sousedů s pravidly a při shodě se akceptuje stav pro konkrétní buňku tak, jak pravidlo definuje. Je-li buňka neaktivní a její oba sousedé jsou aktivní, pak podle pravidla 110 bude buňka v nové generaci aktivní.

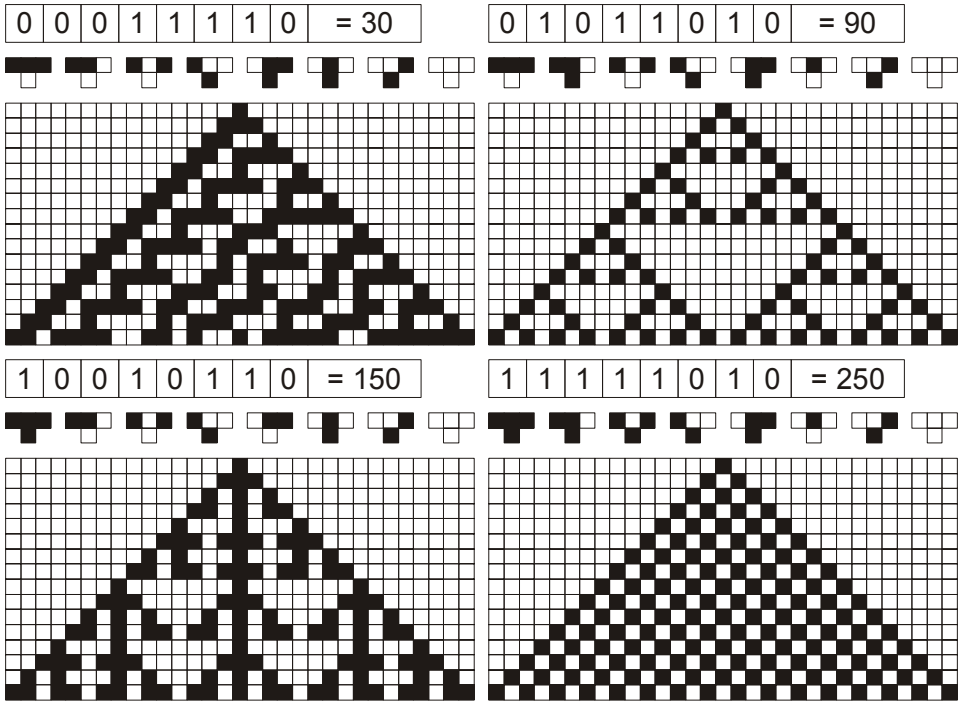
Zápis některých pravidel lze ještě zjednodušit, jako například pravidlo 90



Obr. 3. Pravidlo 90

jež lze vyjádřit pomocí rovnice $a_i^{t+1} = \left(a_{i-1}^t + a_{i+1}^t\right) \bmod 2$, kde t značí iterační krok, i-1, i+1 jsou sousedé buňky a operátor modulo 2 koriguje rozsah opět do binárních hodnot. Tento zápis vede k jednoduššímu výpočtu a tedy urychlení iterací.[4]

Při grafické interpretaci elementárních buněčných automatů se s oblibou využívá rozšíření do dvou-dimenzionálního prostoru, neboť lze takto zachytit několik po sobě jdoucích generací.



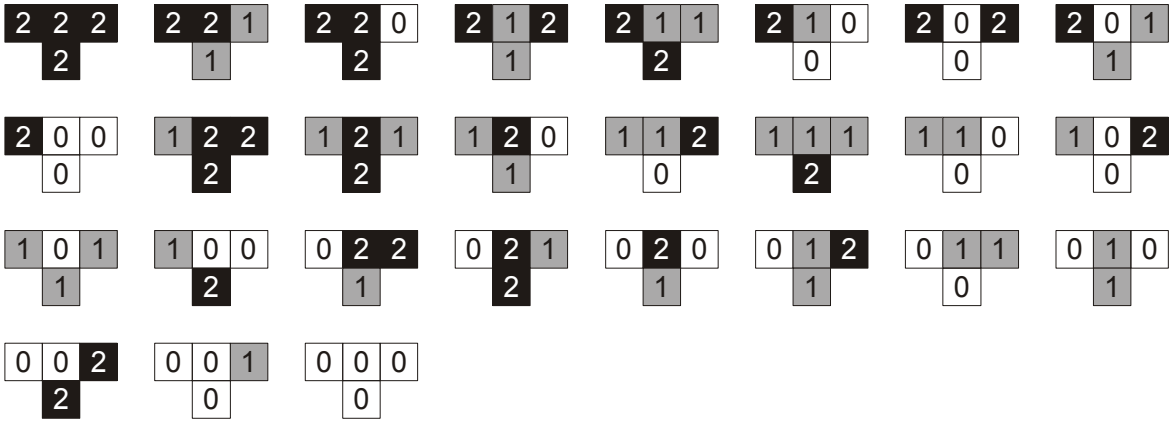
Obr. 4. Grafická implementace generací

2.1.2 Obecné jednorozměrné automaty

Mezi jedno-dimenzionální automaty ale nepatří pouze ty elementární. Je zřejmé, že doposud uvažované parametry automatu jsou velmi limitující. Při jejich rozšiřování má smysl uvažovat jednak o zvýšení počtu možných stavů samotné buňky, tak rozšíření okolí, které bude bráno v potaz při vyhodnocování stavu v nové generaci. Takto bohatý prostor pro volbu parametrů umožňuje vytvořit nekonečně velkou množinu jedno-dimenzionálních automatů což ovšem klade nemalé požadavky na systém zápisu pravidel. Ze zápisu musí být patrné, jakého počtu stavů může buňka nabývat a jak velké je okolí, které hraje roli při vyhodnocování.[9]

Za předpokladu, že okolí bude shodně velké na obou stranách je možná jednoznačná definice jedinou cifrou a celý systém zápisu pak může vycházet z již známého konceptu užitého u elementárních automatů.

Například konstrukce zápisu pravidla pro buňku, za jejíž sousedy se považují všechny do vzdálenosti 1, buňka má tedy 2 sousedy, jednoho vlevo a jednoho vpravo, mající tři možné stavy (0, 1, 2) bude následující.



Obr. 5. Konstrukce obecných pravidel

Výsledné pravidlo se potom tedy zapíše takto:

1/212120010221020012121101200

kde lomítkem oddělená jednička značí velikost okolí, z nejvyššího čísla v pravidle se jednoduše odvodí počet stavů buňky. Je dobré poznamenat, že při počtu stavů nad deset je nutné označení písmeny (10=A, 11=B, …) neboť pozice číslice v pravidle je přesně dána a nelze tedy bez jasného vyznačení užít dvou a víceciferných čísel v pravidle. Za upozornění také stojí skutečnost, že pravidla pro tyto automaty nelze nikterak decimálně zkracovat. Zápis pravidla 110 z příkladu výše má ekvivalent 1/01101110.

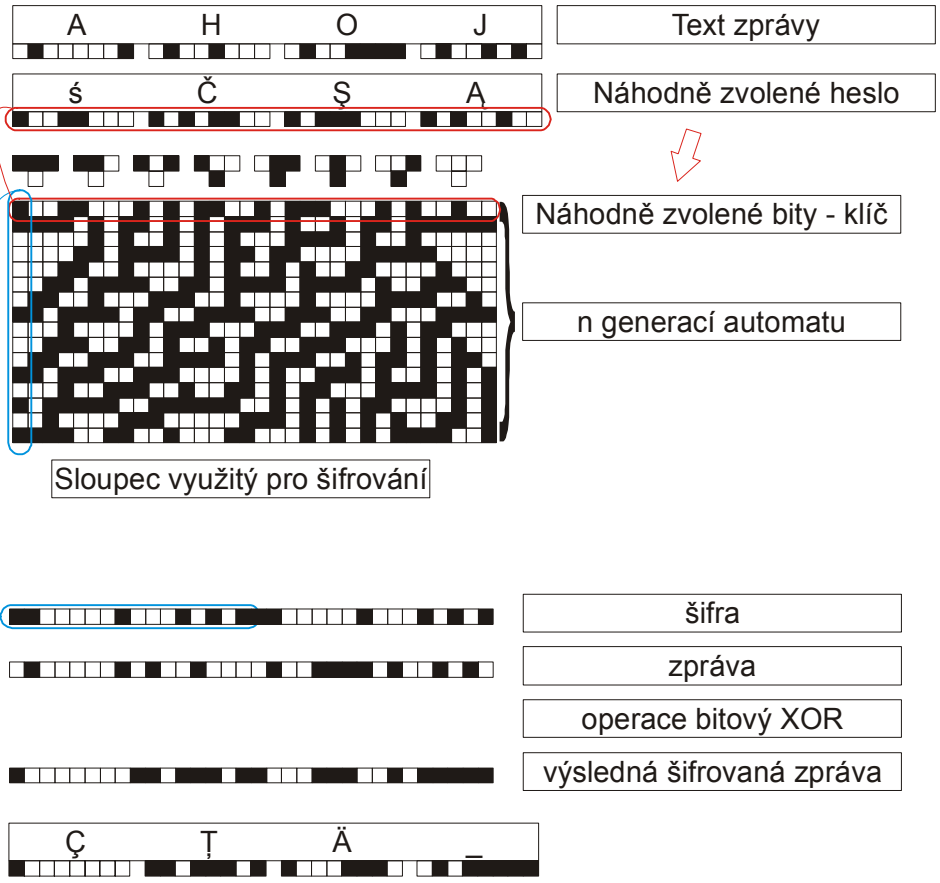
Původní předpoklad symetrického okolí zdánlivě ubírá na obecnosti, Ve skutečnosti je pouze nutné si uvědomit, že z jakkoli nekonzistentního okolí lze symetrické jednoduše vytvořit a pravidla snadno přizpůsobit tak aby vyhovovala zadaným podmínkám.

2.1.3 Praktické aplikace jednorozměrných buněčných automatů

Zcela konkrétní praktickou aplikací je generátor náhodných čísel využitý v programu Mathematica pro generování celočíselných náhodných hodnot.

Další zajímavá aplikace, jedno-dimenzionálních buněčných automatů je v oblasti kryptografie. Pro oba zmiňované případy je velice přínosné zejména pravidlo 30, jímž generovaná bitová sekvence splňuje všechny statické testy náhodnosti. Následující ukázka osvětlí principy využití buněčných automatů v oblasti šifrování.[6]

Nejprve se vygeneruje náhodná sekvence bitů. Počet bitů záleží na požadavku bezpečnosti. Obecně se v současnosti považuje 128 bitový klíč za dostatečně bezpečný, při demonstraci však postačí mnohem kratší. Takto vygenerovaná sekvence je jednou ze tří částí klíče. V druhém kroku tuto bitovou sekvenci předložíme k rozvoji buněčnému automatu s vhodným pravidlem, například zmiňované pravidlo 30, jako 0-tou generaci. Je otázkou volby kolik generací bude vygenerováno, jejich počet je druhou částí klíče. Dalším krokem bude ve vygenerovaném dvourozměrném obrazci volba sloupce, který už bude přímo využit k šifrování zprávy. Pořadí sloupce je poslední částí dešifrovacího klíče. Zprávu v binární podobě pomocí vygenerované posloupnosti zašifrujeme například některou z logických funkcí.



Obr. 6. Postup při šifrování pomocí 1D buněčného automatu

Na obrázku je pomocí náhodně zvoleného klíče „śČŞĄ“, zašifrován text „AHOJ“. Je využito šestnácti generací buněčného automatu s pravidlem 90 a použit první sloupec jeho výstupu. Na text a tento sloupec je aplikována binární funkce XOR, která ve výsledku dává zašifrovaný řetězec „ÇŢÄ_“.

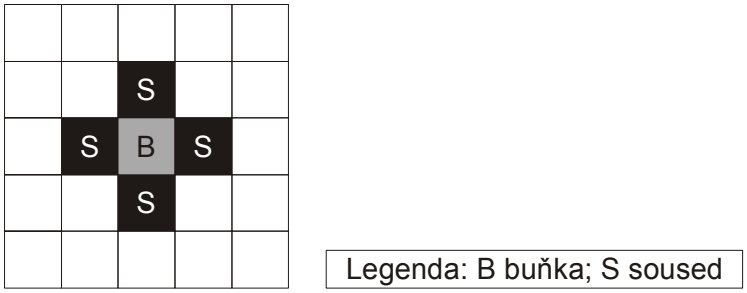
2.2 Dvojdimenzionální buněčné automaty

S přidáním další, tedy druhé, dimenze v buněčných automatech přichází logicky nárůst možností, ale také komplikací. Variabilita dvourozměrný buněčných automatů je nesrovnatelně větší proti jednorozměrným, především díky většímu počtu sousedů každé buňky. Nárůst sousedících buněk na dvoj- až čtyřnásobek přímo směřuje k obohacení základny pravidel a v jejím důsledku tedy zvýšení variability systému.

Ač je na první pohled patrná odlišnost dvourozměrných buněčných automatů od těch jednorozměrných, není vhodné zapomínat na rysy obecně shodné. I u dvourozměrných buněčných automatů je základním stavebním kamenem buňka, ať v nejjednodušším případě dvoustavová, nebo v případech komplikovanějších s počtem stavů bez horní hranice. Tvorba nové generace se od jiných automatů také nikterak neliší a tedy stav buňky v budoucí generaci je funkcí stavu buňky samotné a stavů těch v sousedství. Je zřejmé, že s výčtem není nutné dále pokračovat, neboť kompletní seznam shodných vlastností v podstatě znamená popis zcela obecného buněčného automatu. Jelikož je dvourozměrný automat nadmnožinou jednorozměrného, lze v něm najít množství rozšíření. Jak již bylo řečeno, zřejmě nejviditelnější rozdíl se týká sousedů.

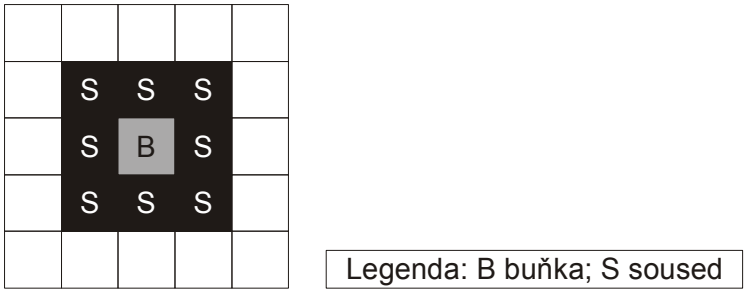
V bezprostředním okolí buňky lze totiž za sousedství považovat různá seskupení.

Snadno představitelná a velmi populární je sousednost čtyř spojitá. Při svých úvahách s ní pracoval von Neumann a po něm také nese i své jméno.



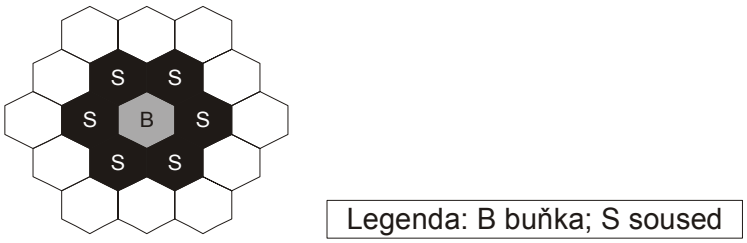
Obr. 7. Neumannova sousednost

Důležitou ve výčtu sousedností je osmi spojitá, Moorova sousednost. Toto pojetí sousednosti využívá například automat „Game of life“

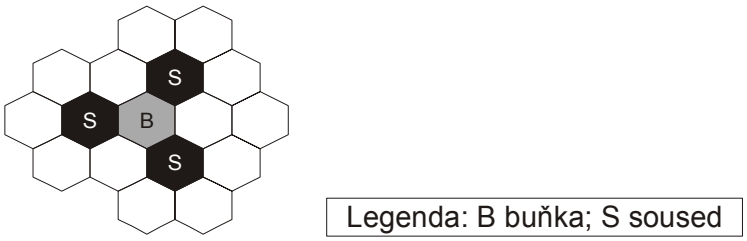


Obr. 8. Moorova sousednost

Další z rozšiřujících možností dvourozměrných buněčných automatů je tvar samotných buněk. Od čtvercových, doposud obvyklých, lze tvarově buňku upravit na pravidelný šestiúhelník. To sebou přináší další z možné řady sousedností, sousednost šesti spojitou, nazývanou hexagonální sousednost, a sousednost tří spojitou, tzv. *tripod* spojitost. Lze zde spatřit s von Neumannovým a Moorovým sousedstvím.

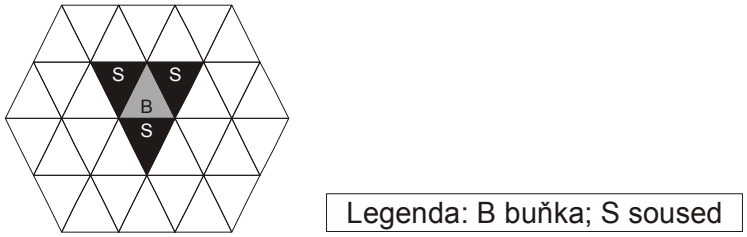


Obr. 9. Hexagonální sousednost



Obr. 10. Tripod sousednost

V případě, že tvarově bude buňka odpovídat trojúhelníku, sousedící buňky budou opět tři (plus středová, že) a podoba sousedů bude následující.[7]



Obr. 11. Trojúhelníková sousednost

Předchozí výčet základních tvarů a sousedností buněk není samozřejmě vyčerpávající. Analogicky lze vytvářet mnohé další.

Jak je vidět, narůstající bohatost automatů nespočívá pouze ve větší variabilitě pravidel, ale už samotné grafické znázornění nabízí rozmanitou škálu různých automatů. V této souvislosti lze vysledovat zajímavý rozdíl při znázorňování automatu. Zatímco u jednorozměrných se generace obvykle znázorňují v ploše a vizuálně se tak zobrazuje nejen nová generace, ale existuje i jakási podoba historie u dvourozměrných se obvykle přistupuje také k zobrazení v ploše, což ovšem umožňuje pouze vizualizaci stavu právě aktuálního. Pro získání obdobné vizuální historie by bylo třeba znázorňovat generace v tří-dimenzionálním prostoru.

Dvourozměrné automaty dále obohacuje velká variabilita pravidel. Jejich klasifikace je vzhledem ke své rozmanitosti také dosti obsáhlá. Počet pravidel, které lze sestavit, uvažujeme-li sousednost von Neumanovu je 2^{32}, u hexagonální sousednosti se nám podaří vytvořit 2^{128} pravidel a u Moorovy sousednosti to už je 2^{512} pravidel. Tím samozřejmě výčet nekončí. V úvahu dále přichází větší okolí, jehož velikost není ničím limitována.

Se stejným problémem se potýká i samotný zápis pravidel, který nelze výrazněji zkrátit bez snížení jejich přesnosti. Typicky lze pravidly pro následující generaci buněk charakterizovat zrod či stagnaci buňky v nové generaci v závislosti na počtu aktivních sousedů. Takové zjednodušení nabízí pravidla semi-totalistická a totalistická. V prvém případě je pro zjištění stavu buňky v nové generaci potřeba sumarizovat počet aktivních buněk v sousedství a na základě tohoto počtu a aktuálního stavu buňky samotné se rozhodne o její následující hodnotě. V případě druhém se do sčítání zahrne i buňka právě korigovaná a rozhoduje se jen na základě absolutního počtu aktivních buněk. Pro semi-totalistická pravidla se využívá zápisu B/S, který před lomítkem udává kolik musí mít buňka sousedů aby se stala z neživé živou a za lomítkem je udán počet pro který buňka

setrvá aktivní i v následující generaci, byla-li aktivní v té současné. Tento zápis je sice velmi přehledný a jednoduchý, ale neumožňuje specifikovat stav buňky v nové generaci na základě zcela konkrétní konfigurace sousedů generace předchozí.

To by bylo možné pouze v případě využití tabulky, kdy každé s možných konfigurací by odpovídal nový stav buňky. Zápis by tak byl kompletní, ale velmi rozsáhlý, neboť tabulka by měla dva sloupce a počet řádků odpovídající počtu možných konfigurací sousedů.[5][7]

2.2.1 Popis automatů s různými podmínkami zrodu a přežití, praktické využití dvourozměrných automatů

V praxi se využívá vícestavových buněčných automatů například v počítačové grafice. Některé filtry využívají pro odvození nové hodnoty pixelu hodnotu současnou a váhově upravenou hodnotu pixelů okolních, tímto způsobem například fungují rozostřující filtry.

Jinou aplikací na kterou byly buněčné automaty použity je simulace dopravy – automat o dvou řádcích reprezentuje dálnici, jednotlivé buňky pak jedoucí vozy. Podmínky zrodu a přežití jsou zde diametrálně odlišné, což jen poukazuje na nepřeberné množství pravidel buněčných automatů a jejich využití.[8]

2.2.2 Hra Život

Hra Život, v originále game of Life, je bezesporu nejznámějším příkladem buněčného automatu. Automat prezentoval britský matematik John Horton Conway v roce 1970. Z teoretického hlediska je hře Život přisuzována stejná výpočetní síla jako univerzálnímu Turingově stroji, jež je schopen vypočíst jakýkoli algoritmicky zapsatelný problém.

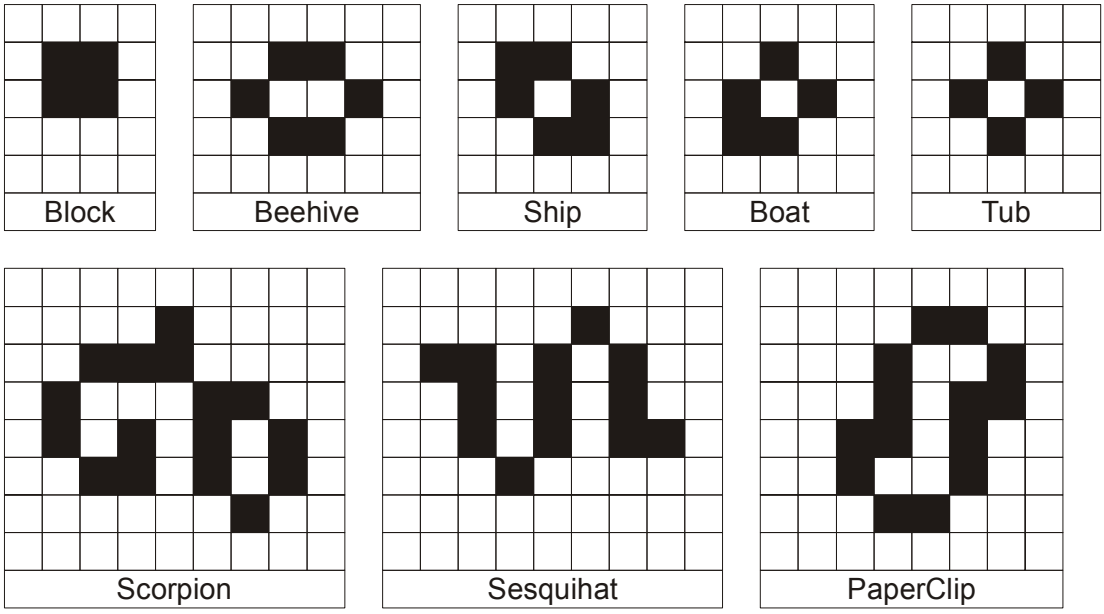
Ačkoli pravidla pro hru Život jsou velmi jednoduchá, nebylo snadné je definovat. Conway vybral pravidla až na základě četných pokusů, kdy eliminoval přílišnou „plodnost“ populace a také její rychlé vymírání.

Za svou popularitu vděčí tento automat především Martinu Gardnerovi, který jej roku 1970 ve svém článku Mathematical Game v časopise Scientific American poprvé představil širší veřejnosti. Automat si získal pozornost mnoha badatelů a také díky nástupu levných počítačů se stal oblíbenou programátorskou úlohou. Zájem vzbudil také v okruhu biologů, matematiků, ekonomů a v neposlední řadě i filozofů.[5]

Ač samotný název hovoří o hře Život, je tento mírně zavádějící. Ve své podstatě se nejedná o hru v pravém slova smyslu, neboť v ní proti sobě nenastupují hráči, nemá vítěze ani poražené. Název lze spíše chápat v přeneseném významu jako analogii „životního cyklu“. Ve své základní podobě odpovídá hra Život specifikům dvourozměrného buněčného automatu s nekonečnou čtvercovou mřížkou. Každá z buněk má dva stavy, je-li buňka naživu, aktivní, potom má hodnotu jedna, případně je vybarvená. Je-li buňka mrtvá, neaktivní, nabývá hodnoty nula, eventuálně ztrácí své zbarvení. Pravidla jsou odvozena od počtu žijících sousedů, přičemž sousednost se zde uvažuje osmi spojitá, tedy Moorova sousednost. Zcela konkrétně pravidla říkají, že buňka v další generaci přežívá, pokud má ve svém sousedství dvě nebo tři aktivní buňky, narodí se, pokud má právě tři žijící sousedy a zahyne ve všech ostatních případech. Pravidla se píší ve zkrácené podobě 23/3, kde číslice před lomítkem specifikují, kdy buňka setrvá aktivní a číslice za lomítkem určuje, při jakém počtu aktivních sousedů se zrodí.

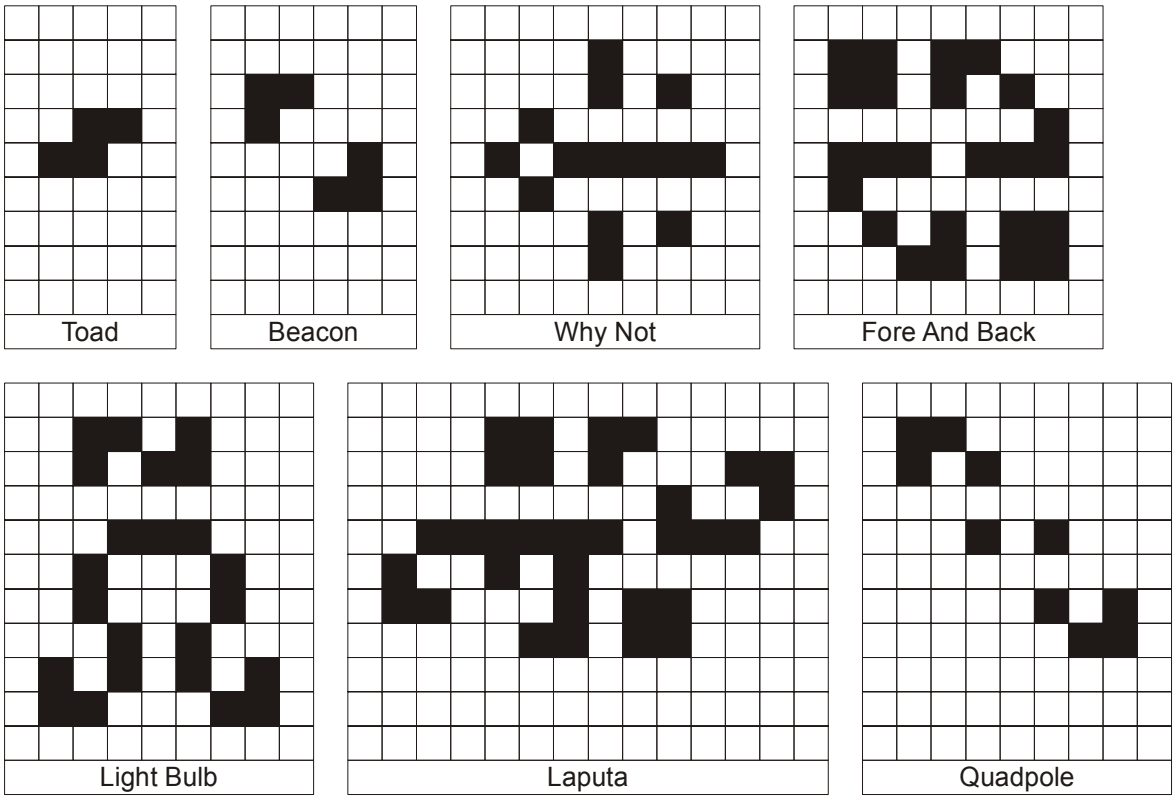
I při takto jednoduchých pravidlech jsou variace vznikajících útvarů nepřeberné. Původně chaotické uspořádání žijících buněk mnohdy vede díky samoorganizaci na uspořádané útvary z nichž lze vyčlenit dvě velké skupiny. Jedna skupina představuje útvary, jež v každém cyklu mění svou podobu a po určitém počtu iterací zcela zaniknou. Ta druhá představuje útvary statické nebo periodické, ty jsou imunní k počtu generací. V případě, že nejsou narušeny, setrvávají ve své konfiguraci neustále naživu. Neustále žijící konfigurace lze dle jejich charakteru ještě dále rozdělit do několika podskupin.

První podskupinou jsou takové útvary, které mají v každé generaci zcela neměnnou podobu:



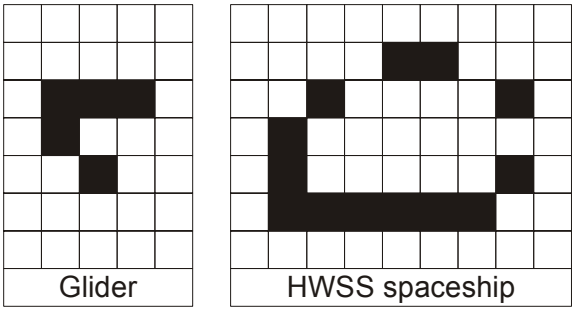
Obr. 12. Statické útvary

Druhou skupinu tvoří takzvané oscilátory. Jsou to objekty, které se s pravidelnou periodou přes několik stavů vrací do původní konfigurace.



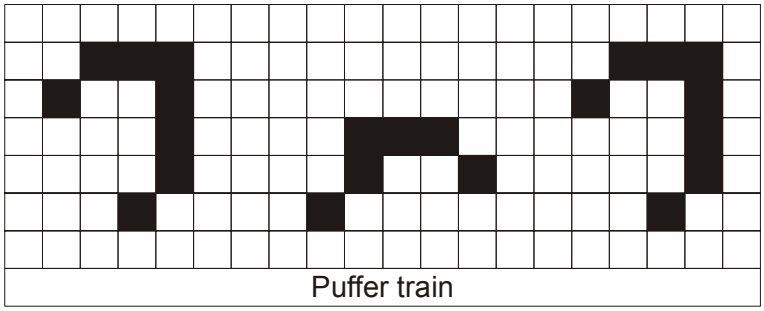
Obr. 13. Oscilátory

Třetí podskupinu tvoří útvary, které se v nějakém směru po ploše pohybují. Patří mezi ně spaceships (vesmírné lodě) s ortogonálním pohybem a gliders (kluzák) s pohybem diagonálním.



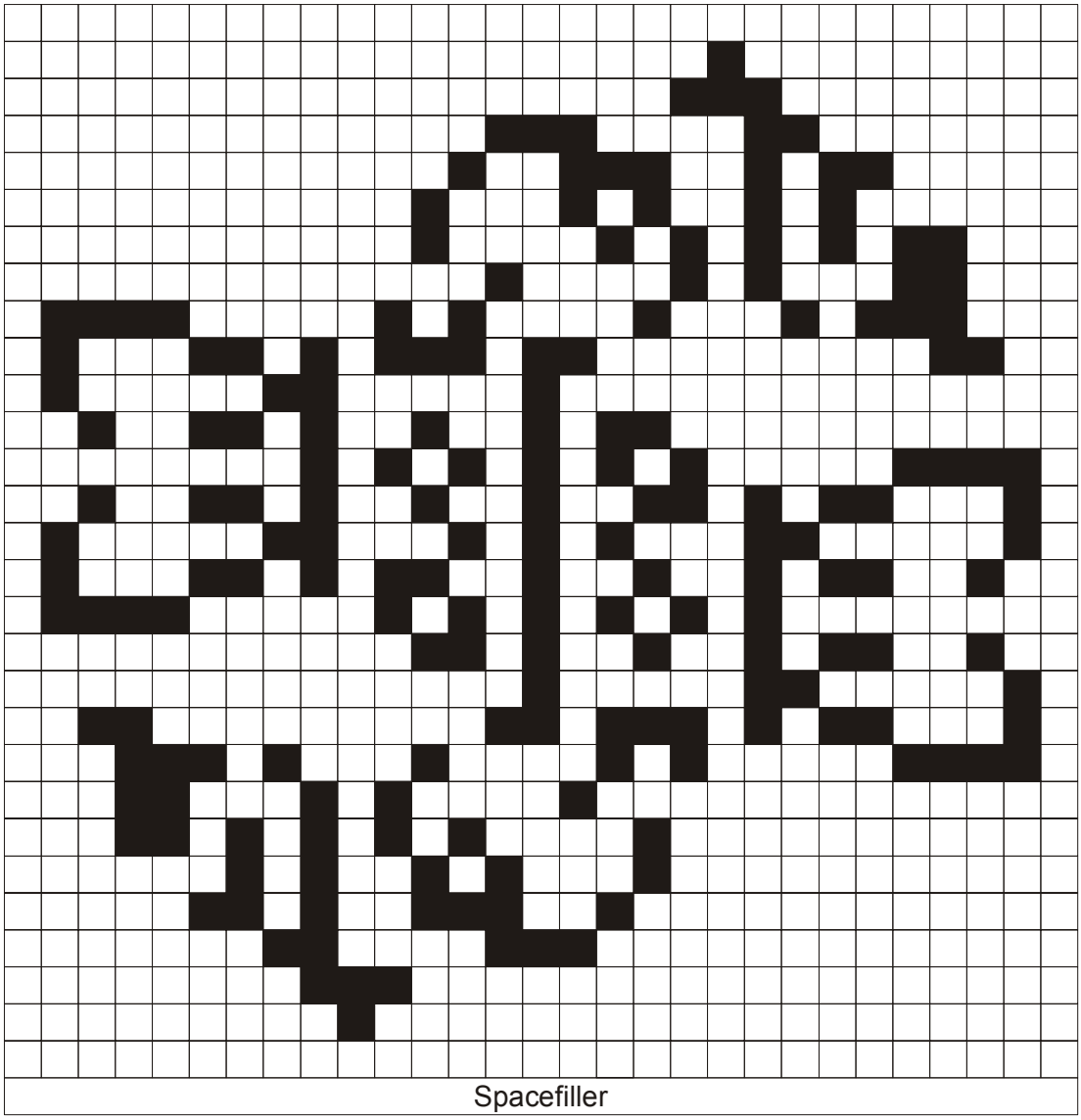
Obr. 14. Pohybující se útvary

Puffer je také útvar, který se v prostoru pohybuje obdobně jako spaceship, rozdílem je skutečnost, že při svém pohybu za sebou zanechává další žijící buňky. Prvním známý útvar tohoto druhu byl objeven Billem Gosperem kolem roku 1971.



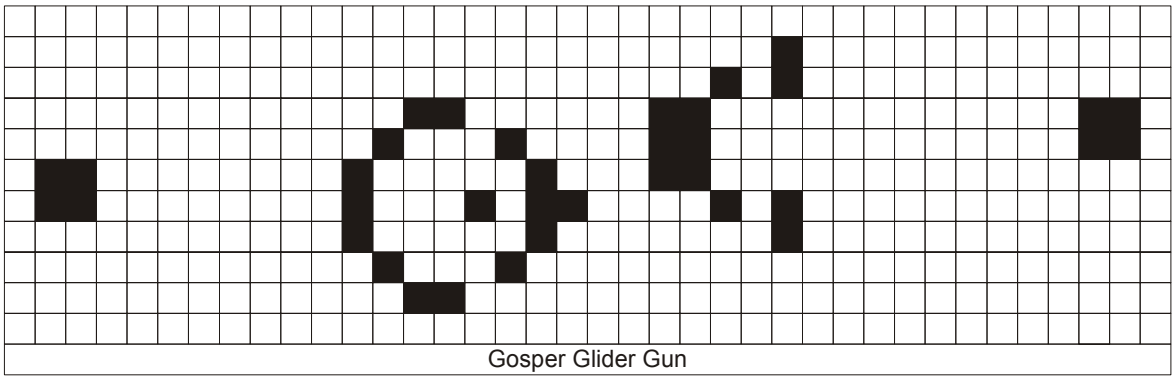
Obr. 15. Puffer

Spacefiller je jakýkoli útvar, který se s kvadratickou rychlostí rozrůstá do prostoru, který zaplňuje. První příklad takového útvaru byl nalezen Hartmutem Holzwartem v roce 1993.



Obr. 16. Spacefiller

Velmi specifické jsou konfigurace takzvaných děl, kteří v pravidelných cyklech chrlí do prostoru pomyslné střely.[10][5]



Obr. 17. Dělo

Chování všech výše jmenovaných útvarů je podmíněno užitím klasického pravidla pro hru Život, tedy 23/3. V průběhu let se však objevily další desítky pravidel, které „hru Život" přizpůsobují k simulaci některých dějů či přírodních úkazů.

Mezi ty nejzajímavější patří:

/3 pravidlo, v jehož důsledku dochází velmi rychle k vymírání populace

/2 pravidlo, díky kterému lze dosáhnou vysokého počtu generací aniž by buňky vymřely. Počet buněk v generaci se po určité době ustálí a v dalších generacích jich výrazně nepřibývá, ale ani neubývá.

23/36 pravidlo vede k většímu počtu zrozených buněk než v klasické hře Život, ale jinak je chování obdobné.

4567/35678 pravidlo vedoucí k rychlému rozrůstání populace

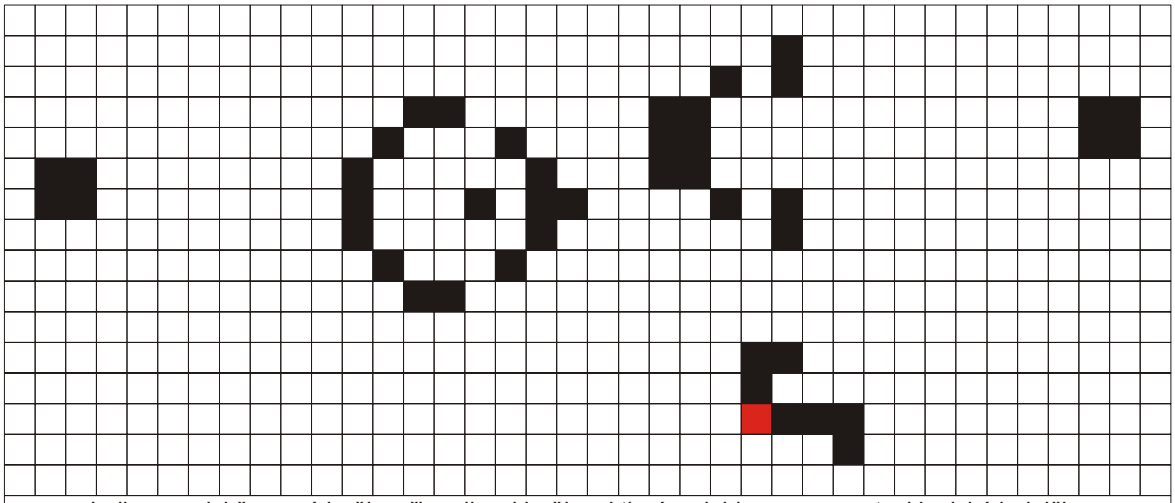
5/346 pravidlo vede na rychlý vznik oscilátorů[5][7][10]

2.2.3 Logické funkce a hra Život

Hra Život je univerzální výpočetní nástroj obdobný Turingovu stroji a proto ji lze mimo simulací některých chemických či biologických jevů využít také k výpočtu například logických funkcí.

Implementovat logické funkce v buněčném automatu především znamená nějakým způsobem reprezentovat proměnné. Možností je například reprezentovat každou proměnnou pomocí útvaru zvaného dělo. Tento útvar v pravidelném cyklu neustále chrlí

do prostoru pomyslné střely, kluzáky, které lze v tomto kontextu považovat za aktivní výstup proměnné. Pokud tedy proměnná nabývá logické hodnoty jedna, potom v implementaci buněčného automatu, dělo reprezentující tuto proměnnou vysílá střely do prostoru. K tomu, aby bylo možné pro proměnnou stanovit i stav logické 0, je za potřebí obohatit základní reprezentaci v buněčném automatu ještě o jednu nezbytnou část. Na obrázku je tedy již kompletní reprezentace proměnné logické funkce v buněčném automatu.

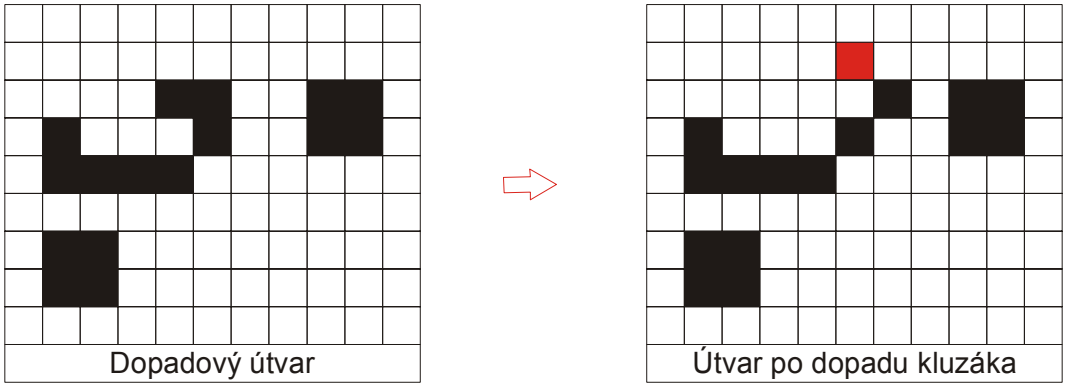


Obr. 18. Implementace proměnné logické funkce v buněčném automatu

Červená buňka umožňuje nastavit, je-li výstup proměnné jedničkový, či nulový. Je-li na místě této buňky buňka aktivní, potom jde o logickou jedničku, neboť první kluzák, který je dělem uvolněn způsobí zaniknutí této části útvaru a další tak již můžou pokračovat do prostoru. Je-li naopak na tomto místě buňka neaktivní, veškerá produkce děla se o tento útvar roztříští a výstup je tímto nulový.

U realizace logické funkce je využito i samotných děl bez přídavků dalších buněk. Tato děla nereprezentují proměnnou, ale jsou nezbytnou součástí při implementaci operátorů.

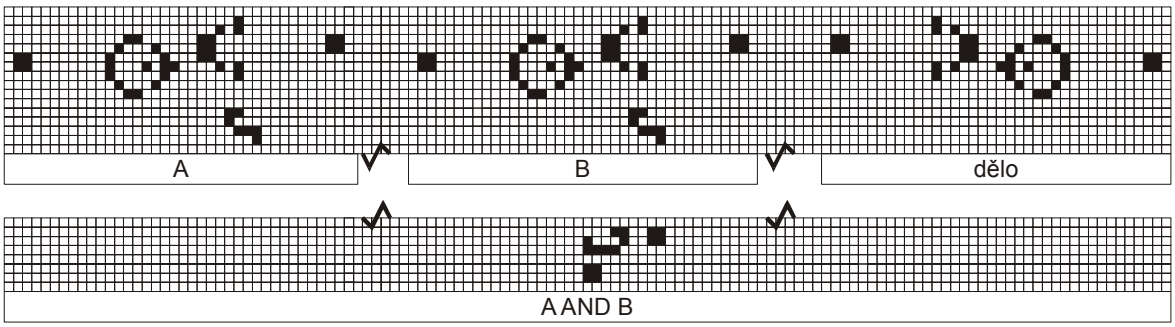
Poslední součástí při realizaci je útvar který reprezentuje výstup.



Obr. 19. Implementace vyhodnocení funkce, výstupu

V případě, že na tento útvar dopadne kluzák, změní se a buňka naznačená červenou barvou reprezentuje výstup, tedy, v případě jedničkového výstupu je buňka aktivní. Pokud kluzák nedopadne, buňka zůstává neaktivní a výstup poté tedy nulový.

Pokud chceme tedy realizovat logickou funkci například A AND B, potřebujeme, dvě proměnné, jedno dělo a dopadovou část. Dělo chrlí kluzáky proti proměnným a v případě, že jsou obě proměnné jedničkové, jedna proměnná eliminuje kluzáky děla a kluzáky druhé proměnné dopadnou na dopadový útvar a vytvoří jedničkový výstup. V případě, že některá z proměnných je nulová, dělo eliminuje kluzáky druhé proměnné a dopadový útvar zůstane nezměněn, tedy s výstupem nula.



Obr. 20. Realizace logické funkce AND

Na obrázku jsou obě proměnné s logickou hodnotou nula, kluzáky děla tedy projdou celou plochou, ale dopadové části se vyhnou a tedy i výstup bude nulový. Proto, aby i kluzáky děla byly eliminovány, přidává se ještě jeden eliminující prvek do míst, kde již neovlivňují danou funkci, ale na samotnou funkčnost to nemá vliv.[11]

Skládáním několika málo prvků, které byly výše jmenovány lze v buněčném automatu realizovat libovolnou logickou funkci.

II. PRAKTICKÁ ČÁST

3 IMPLEMENTACE HRY ŽIVOT

Realizace algoritmu hry Život byla provedena v programu Mathematica. Ve dvou samostatných částech je řešena úloha pro dva a pro tři rozměry.

Z uživatelského pohledu je důležitá možnost nastavení parametrů, které přímo ovlivňují běh hry Život. V programu jsou implementovány proměnné, jejichž nastavením lze ovlivnit velikost prostoru vymezeného automatu, podmínky zrodu a přežití, okrajové podmínky jakož i počet generací, který je pro danou chvíli relevantní.

Mimo nastavitelné parametry programu, je možné zvolit některou z možností generování počáteční generace buněk. Základní možností je náhodné rozmístění. Tato možnost je ve dvourozměrné části dále rozšířena o možnost výběru z přednastavených útvarů. Seznam připravených útvarů lze dále rozšiřovat a tedy vygenerovat si vlastní startovací generaci.

Jako výstup je uživateli prezentováno grafické zpracování podoby jednotlivých generací, které lze projít buď po jednotlivých generacích, nebo si prohlédnout celý běh v animaci.

3.1 Dvourozměrný prostor

3.1.1 Inicializační část

Kód:

```
1 K=50;
2 SeedRandom[];
3 stay={2,3};
4 born={3};
5 toroide=0;
6 pocetgeneraci=50;
7 gol=Table[0,{K+2},{K+2}];
8 utvary={{{0,0},{0,1},{1,0},{1,1}},{{0,1},{0,2},{1,0},{1,3},{2,1},{2,
  2}},{{0,5},{1,1},{1,2},{1,4},{1,6},{2,0},{2,2},{2,4},{2,6},{3,0},{3,
  4},{3,5},{4,1},{4,2},{4,3},{5,3}},{{0,0},{0,1},{1,1},{1,2}},{{0,2},{
  0,3},{1,3},{2,0},{3,0},{3,1}},{{0,2},{0,3},{0,5},{0,6},{1,1},{1,3},{
  1,5},{1,6},{2,0},{3,0},{3,1},{3,2},{3,4},{3,5},{3,6},{4,6},{5,0},{5,
  1},{5,3},{5,5},{6,0},{6,1},{6,3},{6,4}},{{0,0},{0,1},{0,5},{0,6},{1,
  0},{1,2},{1,4},{1,6},{2,2},{2,4},{3,1},{3,5},{4,1},{4,5},{5,2},{5,3}
  ,{5,4},{7,1},{7,3},{7,4},{8,1},{8,2},{8,4}},{{0,1},{1,0},{2,0},{2,1}
  ,{2,2}},{{0,0},{0,1},{0,2},{0,3},{0,4},{0,5},{1,0},{1,6},{2,0},{3,1}
  ,{3,6},{4,3},{4,4}},{{0,2},{0,7},{0,16},{1,3},{1,8},{1,11},{1,17},{2
  ,3},{2,8},{2,9},{2,10},{2,17},{3,0},{3,3},{3,14},{3,17},{4,1},{4,2},
  {4,3},{4,15},{4,16},{4,17}}};
```

V inicializační části se nastavují jednotlivé proměnné důležité pro běh samotného algoritmu hry Život.

Proměnná K reprezentuje rozměr prostoru který bude buňkám vymezen. Jde o celočíselnou hodnotu, jejíž velikost záleží především na útvaru, pro který probíhá simulace. Doporučená hodnota se pohybuje kolem 50ti.

V seznamu proměnné stay se nastavují počty sousedů. při kterých buňka přežívá. Obdobně v seznamu born jsou uvedeny hodnoty pro které se buňka v případě odpovídajícího počtu sousedů zrodí.

Proměnná toroide Nastavuje okrajové podmínky. V případě, že je hodnota proměnné 0, potom i okrajové podmínky jsou nulové. Je –li hodnota nastavena na 1 nastaví se okrajové podmínky na jedničku. Poslední možností je nastavení proměnné na 2, v tomto případě se pomyslná plocha spojí v toroid a pro buňky bude platit, že sousedem buňky zcela vlevo je buňka zcela vpravo, totéž platí i opačně a pro horní a spodní řadu buněk.

V poli utvary jsou seskupeny jednotlivé předdefinované útvary. A to v tomto pořadí:

1 – Block, 2 – Beehive, 3 – Scorpion, 4 – Toad, 5 – Beacon, 6 – Fore And Back, 7 – Light Bulb, 8 – Glider, 9 – Spaceship, 10 – Puffer

3.1.2 Část nastavení první generace

Kód:

```
 9 VlozUtvar[moveX_,moveY_,utvar_]:=Module[{i,k},
10       k=utvary[[utvar]];
11       Do[gol[[k[[i,1]]+moveX,k[[i,2]]+moveY]]=1,
12         {i,Length[k]}];];
13 randomsett[]:=Module[{},
14       Do[Do[
15           gol[[i+1,j+1]]=Random[Integer,{0,1}],
16           {i,K}],{j,K}]];
```

Dvě funkce zajišťující nastavení počáteční generace. V prvém případě, funkce VlozUtvar, je nastavena první generace dle předvolených útvarů, implicitně deset útvarů, a každý z útvarů je možno vložit kamkoli na plochu. V případě druhém se jedná o náhodné rozložení buněk ve startovací generaci

3.1.3 Funkce implementující algoritmus hry Život

Kód:

```
17 Prubeh2D[vstup_]:=Module[{pole2d,pole2dhlp,i,j,a,b,one,soused},
18       pole2d=Table[0,{K+2},{K+2}];
19       pole2dhlp=Table[0,{K+2},{K+2}];
20       pole2d=vstup;
21       pole2dhlp=pole2d;
22       If[toroide==1,Do[pole2d[[1,i]]=1;
23           pole2d[[K+2,i]]=1;
24           pole2d[[i,1]]=1;
25           pole2d[[i,K+2]]=1,{i,K+2}],];
26 If[toroide==2,pole2d[[1,1]]=pole2d[[K+1,K+1]];pole2d[[1,K+2]]=pole2d[
   [K+1,2]];
27     pole2d[[K+2,1]]=pole2d[[2,K+1]];pole2d[[K+2,K+2]]=pole2d[[2,2]];
28         Do[pole2d[[1,i]]=pole2d[[K+1,i]];
29           pole2d[[K+2,i]]=pole2d[[2,i]];
30           pole2d[[i,1]]=pole2d[[i,K+1]];
31           pole2d[[i,K+2]]=pole2d[[i,2]],{i,2,K+1}],];
32       Do[Do[soused=0;
33 Do[Do[If[1==pole2d[[a,b]],If[(((i+1)==a)&&((j+1)==b)),,soused=soused+1
   ],],{a,i,i+2}],{b,j,j+2}];
34           one=0;
35 Do[If[one==0,If[((soused==Part[born,k])&&(pole2d[[i+1,j+1]]==0)),pole2
   dhlp[[i+1,j+1]]=1;one=1,],],{k,Length[born]}];
36 Do[If[one==0,If[((soused==Part[stay,k])&&(pole2d[[i+1,j+1]]==1)),pole2
   dhlp[[i+1,j+1]]=1;one=1,],],{k,Length[stay]}];
37           If[one==0,pole2dhlp[[i+1,j+1]]=0,];,{i,K}],{j,K}];
38       pole2d=pole2dhlp;
39       Return[pole2d];];
```

Funkce Prubeh2D je implementací samotného algoritmu hry Život. Tato funkce je volána pro každou generaci a vygeneruje generaci novou. Vstupem je tedy matice buněk a výstupem opět matice buněk. Načtené pole je přiřazeno do dvojice vnitřních proměnných. Pole se kterým se pracuje je rozšířeno na každou stranu o jednu řadu buněk, což usnadňuje výpočet sousedů. Tyto okrajové buňky jsou dle proměnné toroide nastaveny na příslušnou hodnotu, tedy, pro nulovou proměnnou na nulu, pro jedničkovou na jedničku a pro hodnotu dva se do okrajových částí nakopírují hodnoty příslušných buněk z okrajů původního automatu. Následující cyklus prochází okolí každé buňky a počítá aktivní sousedy. Na základě jejich počtu vyhodnotí pravidla a vygeneruje hodnotu buňky v nové

generaci.Pro okrajové buňky v rozšířeném poli se nové hodnoty nepočítají. Funkce potom vrací novou matici buněk.

3.1.4 Hlavní část

Kód:

```
40 VlozUtvar[10,10,10];
41 maticeMatic=NestList[Prubeh2D,gol,pocetgeneraci];
42 Table[ListDensityPlot[maticeMatic[[i]],Mesh→False,PlotLabel→{"Genera
   ce "i}],{i,1,Length[maticeMatic]}];
```

Poslední část zprostředkuje příslušný počet volání funkce Prubeh2D . Každou z generací která je na výstupu potom uloží a v grafické podobě ji prezentuje jako výsledek běhu programu.

3.2 Třírozměrný prostor

3.2.1 Inicializační část

Kód:

```
 1 K=10;
 2 SeedRandom[];
 3 stay={4,5,6,7,8};
 4 born={6,7,8};
 5 toroide=0;
 6 pocetgeneraci=20;
 7 gol3d=Table[0,{K+2},{K+2},{K+2}];
 8 Do[
 9     Do[
10       Do[
11         gol3d[[i+1,j+1,k+1]]=Random[Integer],
12         {i,K}],
13       {j,K}],
14     {k,K}];
15 <<Graphics`Graphics3D`
```

Obdobná dvourozměrné části, na rozdíl od ní, je zde přímo funkce pro náhodné rozložení buněk v počáteční generaci. Přibývá zavedení modulu pro grafický výstup z důvodů implementace výsledku.

3.2.2 Funkce implementující algoritmus pro 3D

Kód:

```
16 Prubeh3D[vstup_]:=Module[{pole3d,pole3dhlp,i,j,k,a,b,c,one,soused},
17       pole3d=Table[0,{K+2},{K+2},{K+2}];
18       pole3dhlp=Table[0,{K+2},{K+2},{K+2}];
19       pole3d=vstup;
20       pole3dhlp=pole3d;
21       If[toroide==1,
22         Do[
23           Do[
24             pole3d[[1,i,j]]=1;
25             pole3d[[K+2,i,j]]=1;
26             pole3d[[i,1,j]]=1;
27             pole3d[[i,K+2,j]]=1;
28             pole3d[[i,j,1]]=1;
29             pole3d[[i,j,K+2]]=1
30             ,{i,K+2}]
31           ,{j,K+2}],];
32       If[toroide==2,
33         Do[
34 pole3d[[i,1,1]]=pole3d[[i,K+1,K+1]];pole3d[[i,1,K+2]]=pole3d[[i,K+1,
   2]];pole3d[[i,K+2,1]]=pole3d[[i,2,K+1]];pole3d[[i,K+2,K+2]]=pole3d[[
   i,2,2]];
35 pole3d[[1,i,1]]=pole3d[[K+1,i,K+1]];pole3d[[1,i,K+2]]=pole3d[[K+1,i,
   2]];pole3d[[K+2,i,1]]=pole3d[[2,i,K+1]];pole3d[[K+2,i,K+2]]=pole3d[[
   2,i,2]];
36           pole3d[[1,1,i]]=pole3d[[K+1,K+1,i]];
37           pole3d[[1,K+2,i]]=pole3d[[K+1,2,i]];
38           pole3d[[K+2,1,i]]=pole3d[[2,K+1,i]];
39           pole3d[[K+2,K+2,i]]=pole3d[[2,2,i]];
40           ,{i,2,K+1}];
41         Do[
42           Do[
43             pole3d[[1,i,j]]=pole3d[[K+1,i,j]];
44             pole3d[[K+2,i,j]]=pole3d[[2,i,j]];
45             pole3d[[i,1,j]]=pole3d[[i,K+1,j]];
```

```
46              pole3d[[i,K+2,j]]=pole3d[[i,2,j]];
47              pole3d[[i,j,1]]=pole3d[[i,j,K+1]];
48              pole3d[[i,j,K+2]]=pole3d[[i,j,2]];
49              ,{i,2,K+1}],{j,2,K+1}];
50
51          pole3d[[1,1,1]]=pole3d[[K+1,K+1,K+1]];
52          pole3d[[1,K+2,1]]=pole3d[[K+1,2,K+1]];
53          pole3d[[K+2,1,1]]=pole3d[[2,K+1,K+1]];
54          pole3d[[K+2,K+2,1]]=pole3d[[2,2,K+1]];
55          pole3d[[1,1,K+2]]=pole3d[[K+1,K+1,2]];
56          pole3d[[1,K+2,K+2]]=pole3d[[K+1,2,2]];
57          pole3d[[K+2,1,K+2]]=pole3d[[2,K+1,2]];
58          pole3d[[K+2,K+2,K+2]]=pole3d[[2,2,2]]
59          ,];
60        Do[
61          Do[
62            Do[
63              soused=0;
64              Do[
65                Do[
66                  Do[
67  If[1==pole3d[[a,b,c]],If[(((i+1)==a)&&((j+1)==b)&&((k+1==c))),,soused=s
    oused+1],],{a,i,i+2}],{b,j,j+2}],{c,k,k+2}];
68              one=0;
69  Do[If[one==0,If[((soused==Part[born,l])&&(pole3d[[i+1,j+1,k+1]]==0)),p
    ole3dhlp[[i+1,j+1,k+1]]=1;one=1,],],{l,Length[born]}];
70  Do[If[one==0,If[((soused==Part[stay,l])&&(pole3d[[i+1,j+1,k+1]]==1)),p
    ole3dhlp[[i+1,j+1,k+1]]=1;one=1,],],{l,Length[stay]}];
71              If[one==0,pole3dhlp[[i+1,j+1,k+1]]=0,];
72              ,{i,K}],
73            {j,K}],
74          {k,K}];
75        pole3d=pole3dhlp;
76        Return[pole3d];];
```

Naprosto analogická část jako u dvourozměrného automatu.

3.2.3 Generování souřadnic pro výstup

Kód:

```
77  Souradnice[ceho_]:=Module[
78      {i,j,k,poles},
```

```
79      poles={};
80      For[i=2,i≤K+1,
81        For[j=2,j≤K+1,
82          For[k=2,k≤K+1,
83            If[ceho[[i,j,k]]==1,poles=Join[poles,{{i-1,j-1,k-1}}]];
84            k++;];j++;];i++;];
85      Return[poles]; ]
```

Funkce volaná ve chvíli vykreslování generace buněk, prochází matici buněčného automatu a do pole přiřazuje pozice živých buněk. Na této pozici je buňka následně vykreslena. Zádrhel nastává v případě, že žádná buňka v generaci není živá. V tomto případě se totiž do seznamu souřadnic žádná nepřiřadí a díky tomu se také nic nevykreslí. To vede k tomu, že v případě zániků populace se vykreslí jen tolik generací kolik jich obsahovalo živé buňky.

3.2.4 Hlavní část 3D

Kód:

```
86 allgol3d=NestList[Prubeh3D,gol3d,pocetgeneraci];
87 able[ScatterPlot3D[Souradnice[allgol3d[[i]]],PlotStyle→{PointSize[0.
   05],Hue[.6]},PlotLabel→i,PlotRange→{{0,K+1},{0,K+1},{0,K+1}}],{i,Len
   gth[allgol3d]}];
```

opět naprostá analogie dvourozměrného případu. Opět se vytvoří pole obsahující všechny vygenerované generace a poté se tyto generace vykreslí.

ZÁVĚR

Cílem této práce byla rešerše v oblasti buněčných automatů zaměřené na samoorganizaci. Velká pozornost byla věnována nejznámějšímu buněčnému automatu, hře Život a popisu důležitých útvarů, které se v tomto automatu vyskytují. Nicméně hra Život nebyla jediným automatem, který zde byl popsán. Práce obsahuje přehled některých jedno- a dvourozměrných automatů. V práci nejsou opomíjeny ani automaty s větším okolím a u jednorozměrných bylo použito alternativního zápisu pravidel pro větší okolí a vícestavové buňky. Zcela konkrétní praktické příklady prezentují využití automatu na reálných úlohách.

Úkolem praktické části bylo implementovat v prostředí Mathematica hru Život či jakýkoli jiný semi-totalistický automat ve dvou a třech rozměrech. Program byl rozčleněn na dva podcelky, z nichž každý řeší úlohu v jiném rozměru. V realizaci je zahrnuta jednak možnost volby podmínek zrodu a přežití tak počtu relevantních generací či rozměru celého automatu. Pro hru Život jsou k dispozici některé útvary, které lze umístit kamkoli v prostoru buněčného automatu a následně sledovat průběh vývoje těchto seskupení.

Práce by mohla sloužit jako přehled existujících možností v oblasti buněčných automatů a být zdrojem obecných informací pro konkrétně zaměřené problémy. Setříděný přehled by měl pomoci v rychlé orientaci a prezentované informace a příklady k podnícení zájmu o využití buněčných automatů při řešení rozličných praktických úkolů.

SEZNAM POUŽITÉ LITERATURY

[1] WOLFRAM, Stephen. *A New Kind Of Science* 1. vyd. Wolfram Media, 2002. ISBN 1-57955-008-8.

[2] WOLFRAM, S., Cellular automata as simple self-organizing systems, 1982, Caltech preprint CALT-68-938

[3] WOLFRAM, S., University and complexity in cellular automata, 1984, Physica 10D

[4] WOLFRAM, S., Cellular Automata, 1983, Los Alamos Science

[5] WIKIPEDIA, [cit. 2005-05-30]. Dostupné z WWW: <http://www.wikipedia.org/>

[6] BADR, Amr. An Alternative Cellular Automata Kryptogram, 2002, Fakulty of Computer & Information, Dept. Of Computer Science, Cario University

[7] CELL-AUTO, [cit. 20005-05-30], dostupné z WWW: <http://cell-auto.com/>

[8] SIMON, P.M., GUTOWICZ, H.A., Cellular automaton model for bidirectional traffic, 1998, Phys. Rev. E 57, 2441-2444

[9] STEPNEY, Susan, 2004/2005, Cellular Automata, dostupné z WWW: <http://www-course.cs.york.ac.uk/nsc/>

[10] Lexikon, dostupné z WWW http://www.argentum.freeserve.co.uk/lex_home.htm

[11] LogiCell 1.0,

Dostupné z WWW <http://www.rennard.org/alife/english/logicellgb.html >

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

BA Buněčný Automat

CA Cellular Automata

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH