Středoškolská odborná činnost 2007/2008

obor 01.

Univerzalita v celulárních automatech

Autor:
Robert Šiška
Gymnázium Kroměříž, Masarykovo náměstí 469
Kroměříž 767 01, 2. ročník

Kroměříž, 2008
Zlínský kraj

Univerzalita v celulárních automatech

Robert Šiška
boby_my@mail.com

21. května 2008

Celulární automaty jsou dynamické systémy, které i přes svou vnitřní jednoduchost vytváří velmi složité chování. Toto chování může být rozděleno do několika skupin od velmi pravidelných až po chaotické. Cílem práce je shrnout nejdůležitější a nejzajímavější záležitosti týkající se teorie celulárních automatů a popsat způsob, kterým mohou jednoduché celulární automaty počítat.

Obsah

Seznam obrázků

1 Úvod

1.1 Celulární automaty

Celulární automaty (anglicky Cellular automata, zkráceně CA) jsou decentralizované, prostorově i časově diskrétní systémy. Celulární automat se skládá z buněk a pravidel. Buňka je jednoduché zařízení nesoucí informaci ve svém stavu a jednotlivé buňky jsou určitým způsobem uspořádané v prostoru. Každá buňka je dále schopna zjistit, v jakém stavu se nachází buňky sousední, tj. buňky v určité blízkosti, a na základě toho se rozhodnout, do jakého stavu přejde. Toto rozhodování provádí buňky pravidel, která jsou pro všechny buňky stejná. Celulární automat pracuje tak, že v každém časovém kroku změní buňky svůj stav na základě svého stavu a stavů okolních buněk podle pravidla. Celulární automaty jsou využívány v mnoha různých odvětvích jako fyzika, biologie, chemie, biochemie, geologie a další. Pomocí celulárních automatů bylo úspěšně namodelováno například proudění kapalin, růst krystalů, šíření infekčních nemocí nebo lesních požárů. Nejvíce zkoumány jsou ale z pohledu matematiky a informatiky. I přes jednoduchý koncept mohou celulární automaty vyvíjet velmi rozmanité a složité chování. I když je princip založen na lokální interakci buněk, je potřeba zkoumat globální chování celého systému. Názorné je připodobnění k vodě, jejíž molekuly mohou být stejně jako buňky celulárního automatu ovlivněny pouze okolními molekulami. Globální vlastnosti vody jako vlnění nebo proudění, které vyplývají z tohoto principu, jsou ale velmi složité až chaotické. Andrew Ilachinski ve své knize [18] považuje za nejsložitější systém fungující na principu podobném celulárním automatům lidský mozek, skládající se z lokálně interagujících neuronů a vykazující ohromně složité chování.

1.2 Historie

První systémy podobné celulárním automatům navrhl v roce 1948 známý matematik John von Neumann, jehož záměrem bylo nalezení matematického modelu biologické evoluce. Pro svou práci použil poznatky svého kolegy Stanislawa Ulama, který chtěl napodobit růst krystalů ve čtvercové síti v diskrétním čase. Výsledek von Neumannovi práce byl první celulární automat – univerzální konstruktor, který byl schopen sebereprodukce a byl výpočetně univerzální. Asi o dvacet let později, v roce 1970, se britský matematik John Horton Conway, známý hlavně z oblasti teorie her, snažil nalézt jednodušší model, který by zachovával všechny myšlenky univerzálního konstruktoru. Vymyslel tak nejznámější celulární automat vůbec – Hru života (Game Of Life), která byla až donedávna nejjednodušším výpočetně univerzálním celulárním automatem. V osmdesátých letech začal Stephen Wolfram publikovat články o novém typu celulárních automatů, které splňovaly všechny myšlenky původního von Neumannova návrhu, avšak v jednorozměrném prostoru, na rozdíl od dřívějších dvourozměrných automatů. Wolfram zjistil, že všechny druhy chování dvourozměrných automatů se vyskytují i v jeho jednorozměrných celulárních automatech. Předpověděl univerzalitu základního automatu 110, což v 90. letech formálně dokázal Matthew Cook. V roce 2002 uveřejnil Wolfram kontroverzní knihu A New Kind Of Sci-

ence [8], na které pracoval více než deset let a která je nyní je volně dostupná na Internetu.

1.3 Definice

1.3.1 Konečný automat

Konečný automat $\mathcal{M}$ je pětice $(Q, \Sigma, \delta, q_0, F)$, kde

- Q je neprázdná konečná množina stavů.
- Σ je konečná množina vstupních symbolů, nazývaná také vstupní abeceda.
- $\delta : Q \times \Sigma \to Q$ je parciální přechodová funkce.
- $q_0 \in Q$ je počáteční stav.
- $F \subseteq Q$ je množina koncových (akceptujících) stavů.

Definice převzata z [1]

Konečný automat může být chápán jako jednoduchý stroj obsahující čtecí hlavu, konečnou pásku a konečně stavovou řídící jednotku (paměť). Jeho práce může být jednoduše popsána tak, že automat přijímá symboly patřící do Σ a podle přechodové funkce δ změní svůj stav patřící do množiny Q. Pokud se automat dostane do jednoho ze stavů z množiny F, akceptuje tuto posloupnost symbolů, neboli slovo.

1.3.2 Celulární automat

Celulární automat je potenciálně nekonečná, prostorově i časově diskrétní soustava identických konečných automatů, které jako své vstupní informace přebírají stavy vedlejších konečných automatů a na jejich základě mění svůj stav. Jednotlivé konečné automaty nazýváme buňkami celulárního automatu. Koncept buňky a konečného automatu je tedy shodný, jiný je ale účel. Zatímco úkol konečného automatu je rozhodovat o slovech, buňka slouží jen jako nositel informace. Celulární automaty von Neumannova typu musí splňovat tyto vlastnosti:

- **Diskrétnost prostoru**.
- **Diskrétnost stavů:** každá buňka nese jeden z konečné množiny stavů.
- **Diskrétnost času:** v každém časovém kroku vypočítá každá buňka stav, který bude nabývat v dalším časovém kroku.
- **Stejnorodost:** všechny buňky jsou ekvivalentní.
- **Lokální interakce:** každá buňka interaguje pouze se svými sousedními buňkami.

Definice

Celulární automat $\mathcal{A}$ je čtveřice (M, Σ, n, δ), kde

- M je diskrétní prostor. Konečný, či nekonečný.
- $\Sigma \in Z^+$ je množina všech možných stavů buňky.
- n je velikost sousedství.
- δ je přechodová funkce.

Prostor M může být vícerozměrný. Počet rozměrů označíme d, pak d určuje také tvar celulárního prostoru ($d = 1$ řada buněk, $d = 2$ čtvercová síť atd.) $k = |\Sigma|$ určuje kolika stavů může buňka nabýt. Musí být splněno, že $k \geq 2$. Velikost sousedství určuje počet okolních buněk, které mají být použity pro přechodovou funkci.

1.3.3 Prostor a konfigurace celulárního automatu

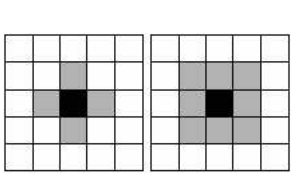
Stav S automatu A je určen posloupností $f_S^A : Z \rightarrow \Sigma$. U konečných automatů je funkce f definována pouze pro hodnoty s indexem $0 \leq i < |M|$. V ostatních případech je $f(i)$ = nedefinovaná hodnota. Stav buňky s indexem i je $S_i = f_S^A(i)$, pokud je automat zřejmý z kontextu, zapisujeme $S_i = f_S(i)$.
Prostor je potenciálně nekonečná d-rozměrná síť souřadnic. Pro $d = 1$ je prostor řada buněk, pro $d = 2$ pole buněk atd. Tato práce se zaměřuje převážně na jednorozměrné ($d = 1$) celulární automaty. Protože v praxi nemůžeme vytvořit nekonečně velký celulární prostor, musíme vyřešit okrajové podmínky. První řešení je takové, že myšlené buňky za okrajem jsou pevně v určitém stavu. Další řešení je takzvané zrcadlové ohraničení – krajní buňky mají stejnou hodnotu jako jejich existující soused. Nejefektivnější a nejpoužívanější řešení je ,,slepit" konce tak, že vedle buňky na levém konci je buňka na pravém konci. Pro jednorozměrné ($d = 1$) automaty tedy vytvoříme kruhový prostor, pro $d = 2$ torus. Pro $d > 2$ je výsledný prostor hůře představitelný, ale stále poměrně jednoduše implementovatelný.

1.3.4 Sousedství a pravidla

Buňka i se v čase t nachází ve stavu $S_i^t \in \Sigma$. **Sousedství** N_i^t je uspořádaná n-tice obsahující sousedy buňky i.

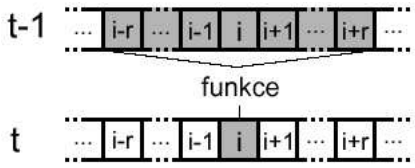
$$N_i^t = (S_{i-r}^t, S_{i-r+1}^t, ..., S_i^t, ..., S_{i+r-1}^t, S_{i+r}^t)$$

kde r je poloměr, $r = \frac{n-1}{2}$. Buňka i je v sousedství N_i^t středová. Poměrně složitější situace je u dvourozměrných automatů, protože sousedství může být reprezentováno více způsoby. Nejpoužívanější dvě formy sousedství jsou von Neumannovo a Moorovo (obrázek 1). První použil von Neumann při návrhu svého univerzálního konstruktoru. Obsahuje všechny buňky, do kterých lze přejít z centrální buňky v r krocích za použití směrů nahoru, dolů, doleva a doprava. Moorovo sousedství je použito v *Game of Life*. Princip r kroků je stejný jako u von Neumannova s použitím dodatečných směrů nahoru+doprava, nahoru+doleva, dolů+doleva, dolů+doprava. Ostatní formy sousedství nejsou příliš používané, nebo jsou použity jen ve specifických případech (např. Billiard-Ball Model v [10]).



Obrázek 1: von Neumannovo a Moorovo sousedství

Pravidlo celulárního automatu je přechodová funkce, která ze sousedství N_i^{t-1} určí stav buňky i v čase t. Přechodová funkce $\delta : \Sigma^n \rightarrow \Sigma$ vrací pro každé možné N_i^t právě jedno S_i^{t+1}.



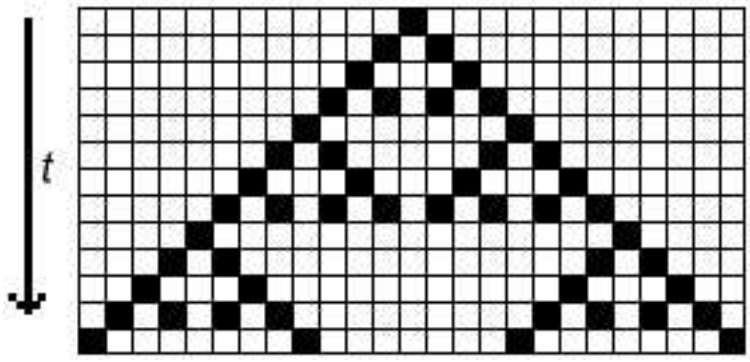
Obrázek 2: Grafické znázornění přechodové funkce

$S_i^t = \delta(N_i^{t-1})$

Jelikož je možných N_i^t konečně mnoho, může být přechodová funkce vyjádřená tabulkou, která každému N_i^t přiřazuje právě jeden prvek patřící do Σ (viz další kapitola).

1.3.5 Časoprostorové diagramy

Nejrozšířenější a nejpraktičtější forma zobrazení celulárních automatů je tzv. časoprostorový diagram (funkce času), který lépe informuje o průběhu celulárního automatu. Tvoří se tak, že zobrazíme konfiguraci automatu v čase $t + 1$ pod nebo nad konfiguraci v čase t. Časoprostorové diagramy jsou praktické hlavně u jednorozměrných automatů. Pokud provedeme ,,slepení" krajových buněk, jak bylo vysvětleno v sekci 1.3.3 je výsledný diagram vlastně plášť válce. Na obrázku 3 je vyobrazeno prvních třináct časových kroků základního automatu 90. Rozměr celulárního prostoru je 25 buněk a v počáteční konfiguraci je pouze jedna buňka ve stavu 1. U dvou a vícerozměrných celulárních automatů se tato technika neuplatňuje, protože by výsledek byl naopak nepřehledný. U dvou a trojrozměrných automatů se tedy časové kroky zobrazují jeden po druhém ve stylu ,,slideshow".



Obrázek 3: Prvních třináct časových kroků základního celulárního automatu č. 90

2 Typy celulárních automatů

Celulární automaty von Neumannova typu se dělí do tříd podle toho, jak přechodová funkce využívá sousedství k rozhodování o dalším stavu.

2.1 Základní

Základní (elementární) celulární automaty byly navrhnuty Stephenem Wolframem v [2]. Je to nejjednodušší třída celulárních automatů. Základní celulární automaty mají dva stavy – každá buňka může být ve stavu 1 nebo 0. Výsledný stav buňky závisí jen na nejbližších sousedech (každá buňka má dvě bezprostředně sousedící buňky – vlevo a vpravo). N_i^t je tedy v tomto případě trojice buněk levá-středová-pravá. Počet všech možných podob sousedství N_i^t je $k^n = 2^3 = 8$. Proto počet všech pravidel je $k^{k^n} = 2^{2^3} = 256$, z nichž každé může být přesně určeno 8bitovým číslem způsobem, který se nazývá Wolframova notace. Následující tabulka určuje pravidlo jednoho základního celulárního automatu. V záhlaví jsou uvedeny všechny případy sousedství, v dolní pak výsledek přechodové funkce pro toto sousedství.

111	110	101	100	011	010	001	000
0	1	1	0	1	1	1	0

$01101110_2 = 110$
Číselné vyjádření tohoto pravidla je tedy 110. Druhý způsob zápisu pravidla je logickou funkcí. Jednotlivé tři buňky v sousedství N_i^t označíme jako a, b, c a můžeme sestrojit logickou funkci tím, že opíšeme případy, ve kterých vrací hodnotu 1 (vycházejme opět z tabulky).

$$F(a,b,c) = ab!c + a!bc + !abc + !ab!c + !a!bc$$

kde '!' znamená negaci a '+' '.' znamenají logický součet a součin. Pomocí Booleovy algebry můžeme formuli zminimalizovat:

$ab!c + a!bc + !abc + !ab!c + !a!bc =$
$ab!c + a!bc + !ab(c + !c) + !a!bc =$
$ab!c + !ab + !bc(a + !a) =$
$b(a!c + !a) + !bc =$
$b(!a + !c) + !bc =$
$b!a + b!c + !bc$

Pokud bychom použili sofistikovanější metody minimalizace, mohli bychom dojít k ještě jednoduššímu zápisu. Pokud za parametry dosadíme buňky ze N_i^t, výsledek funkce bude stav buňky v dalším časovém kroku. Protože je logických funkcí o třech parametrech 256, reprezentuje každá taková logická funkce určitý základní celulární automat.

Stav buňky i v následujícím časovém kroku vypočítáme:

$$S_i^{t+1} = \delta(S_{i-1}^t, S_i^t, S_{i+1}^t)$$

Některá pravidla základních celulárních automatů mají speciální vlastnosti, díky kterým jsou studovány podrobněji. Pravidlo 30 lze použít jako velmi dobrý a jednoduše implementovatelný generátor pseudo-náhodných čísel. K tomuto účelu bylo použito ve slavném programu Mathematica. Pravidlo 90 tvoří známou fraktálovou strukturu Sierpinskiho trojúhelník. Pravidlo 54 je považováno za dalšího kandidáta s univerzálním chováním, nicméně tento předpoklad ještě není dokázán. Pravidlo 110 je výpočetně univerzální (tento pojem a všechny zajímavé vlastnosti tohoto pravidla rozvedeme později v kapitole 3). Přestože je celkový počet pravidel 256, některá pravidla vykazují velmi podobné nebo přímo shodné chování. Každé pravidlo má navíc takzvané zrcadlené pravidlo, které vytváří shodné chování pouze s tím rozdílem, že jsou stavy invertovány, tj. když má v konfiguraci původního automatu buňka stav 1, v zrcadleném automatu bude mít stav 0 a naopak. Zrcadlené pravidlo vytvoříme tak, že původní pravidlo znegujeme a zrcadlově obrátíme. Například pravidlo 01101110_2 (110_{10}) má zrcadlené pravidlo 10001001_2 (137_{10}), které vyvíjí úplně shodné chování.

2.2 Obecné

Obecné automaty fungují na stejných principech jako základní – tedy pro jakékoliv N_i^t speciálně definuje přechodová funkce výstup, konstanty k a r ale můžou nabývat libovolných hodnot. Tento typ automatů se používá hlavně na převod součtových automatů do stavu, na který můžeme použít matematické metody parametrizace a různé zkoumání přechodové funkce. Obecné automaty jsou nadmnožinou všech celulárních automatů. Každý základní, součtový i externě součtový automat může být vyjádřen ekvivalentním obecným automatem.[1] Nejvýznamnější prací využívající obecné automaty je genetické programování celulárních automatů v [11].

[1] Ne však naopak

2.3 Součtové

Součtové automaty nejsou omezeny v žádné konstantě. Rozměr, velikost sousedství i počet stavů mohou být různé. Hlavní rozdíl oproti základním a obecným automatům je v tom, že přechodová funkce nezohledňuje přesné uspořádání buněk, ale pouze součet stavů okolních i středové buňky. Přechodová funkce tedy nepotřebuje jako parametr sousedství N_i^t, ale pouze číslo – součet stavů v N_i^t.

Počet záznamů v tabulce pravidel je tedy menší. Přesněji $(2r+1)k-2r$. Počet všech možných pravidel je $k^{(2r+1)k-2r}$

Nový stav buňky vypočítáme:

$$S_i^{t+1} = \delta(S_{i-r}^t + ... + S_i^t + ... + S_{i+r}^t)$$

Parametr je číslo, které může nabývat hodnoty $\{0, 1, .., (2r+1)(k-1)\}$.

Každé pravidlo může být stejně jako základní automaty popsáno jedním číslem ve Wolframově notaci, už ale trochu složitějším způsobem. Jako příklad si definujeme $k = 3, r = 1$. Záhlaví tabulky obsahuje všechny možné sumy v sousedství, první řádek pak výsledek přechodové funkce pro toto sousedství.

6	5	4	3	2	1	0
0	2	1	1	0	2	0

Neboť máme $k = 3$, výstupní číslo už není ve dvojkové, ale ve trojkové soustavě. Kód pravidla je tedy $0211020_3 = 600_{10}$. Prakticky je Wolframova notace jen převod z určité číselné soustavy se základem k (v tomto případě trojkové) do desítkové, což můžeme provést vzorcem

$$c = \sum_{i=0}^{n} \delta(n-i)k^i$$

kde $n = (2r+1)(k-1)$

2.4 Externě-součtové

Externě-součtové automaty nemají podobně jako součtové žádné omezení konstant. Na rozdíl od součtových se externě-součtové automaty nerozhodují pouze podle součtu stavů buněk v sousedství, ale navíc zohledňují i stav centrální buňky jako samostatný parametr.

Obecný zápis přechodové funkce:

$$S_i^{t+1} = \delta(S_i^t, S_{i-r}^t + ... + S_{i-1}^t + S_{i+1}^t + ... + S_{i+r}^t)$$

Funkce má tedy dva číselné parametry: stav buňky i a součet stavů všech ostatních buněk ze sousedství.

U externě součtových automatů je převod do Wolframovy notace mírně složitější:

$$c = \sum_{j=0}^{k-1} \sum_{i=0}^{n} \delta(j, n-i)k^{ki+j}$$

kde $n = 2rk$

Všimněte si, že v číselném kódu není zakomponována žádná konstanta, proto spolu s číslem musíme udat hodnoty k, r i d.

2.5 Nejstudovanější celulární automaty

Některé celulární automaty se proslavily například pro svou vzhledovou zajímavost, nebo schopností simulovat určité fyzikální děje. Jako nejzajímavější jsem vybral tři: **Game Of Life**, **Wire World** a **Langtonovu smyčku**. Uvádím jen základní informace, neboť o každém z nich už bylo napsáno mnoho publikací a není problém nalézt podrobné informace o každém z nich.

2.5.1 Hra života

Hra života je bezesporu nejstudovanější celulární automat. Je to dvojrozměrný dvoustavový externě-součtový automat, který přes svá extrémně jednoduchá pravidla vykazuje složité, univerzální chování. Ve většině publikacích jsou pravidla vysvětlována slovně:

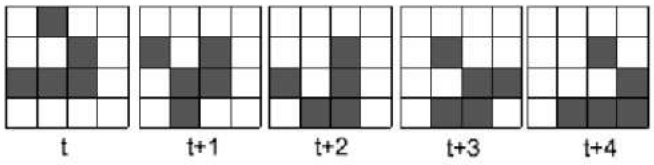
- Pokud je středová buňka rovna 1 a součet jejího osmikolí je 2 nebo 3, pak je nový stav středové buňky opět 1.
- Pokud je středová buňka rovna 0 a součet jejího osmikolí je přesně 3, pak je nový stav 1.
- Ve všech ostatních případech je nový stav buňky 0.

Pro tuto jednoduchost je aplikace tohoto automatu často zadávána jako úloha v učebnicích pro začínající programátory.

Jako důsledek popularity začali lidé definovat pravidla více neformálními výrazy. Buňky ve stavech 0 a 1 bývají označovány jako mrtvé a živé, umírají na přemnožení, osamělost apod. Pro externě součtové binární automaty se také vyvinula speciální notace, která značí, při kolika sousedech má buňka přežít (survive) a při kolika se zrodit (born). Hra života je v této notaci zapsána: S:23/B:3 a používá se téměř ve všech aplikacích namísto klasické Wolframovy notace. [2]

Nejzajímavější na Hře života je existence gliderů a jejich generátorů (glider guns), které je vysílají v určitém směru po určitém počtu časových kroků. Glider je malá skupina buněk, která se pohybuje určitým směrem v prostoru. Hra života je schopna provádět jakýkoliv výpočet tím, že simuluje hradlové součástky. Skupina gliderů svou přítomností resp. nepřítomností vyjadřují bity 1 a 0. Pomocí mnohem složitějších struktur lze sestavit logická hradla AND, OR a NOT, která mohou být zapojena do logických obvodů. To znamená, že lze v tomto automatu vytvořit simulaci digitálního počítače. Vytvoření takovéto simulace v Game Of Life je extrémně složité, nicméně ne nemožné. Navíc se ukázalo, že digitální počítač není jediný univerzální model, který může být simulován, neboť byl v tomto automatu sestrojen Turingův stroj. Navrhl ho a sestavil Paul Rendell v [4].

[2] Jen pro zajímavost, ve Wolframově notaci je číslo tohoto automatu 224.



Obrázek 4: Schéma pohybu nejjednoduššího glideru. V čase t+4 je jeho konfigurace stejná jako v čase t, pouze posunuta.

Kromě gliderů existují v Life také další zajímavé struktury jako bloky, pulsary, vesmírné lodě a podobně. Jejich kompletní galerie je k nalezení dokonce na mnoha místech na Internetu. Zajímavé je, že lidé stále objevují nové struktury, přestože tento automat existuje už přes čtyřicet let. Pro bližší studium doporučuji důkaz o univerzalitě Game of Life v [3] a stránky Paula Rendella, na kterých detailně vysvětluje konstrukci svého Turingova stroje. Autorem Hry života je John Horton Conway, vědec známý převážně z oblasti teorie her.

2.6 Wire World

Možnost simulace digitálního počítače ve Hře života byla objevena dodatečně, ale Wire World byl pro tento účel přímo navržen. Zatímco glidery pouze sledovaly myšlené vodiče u Wire World jsou vodiče přímo vyobrazeny a sestavování logických součástek je o mnoho jednodušší. Wire world je vlastně přímá simulace digitálního počítače.

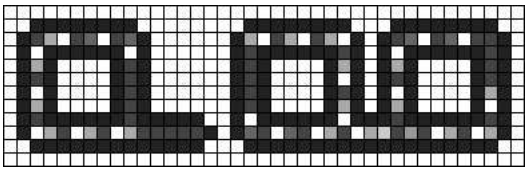
Stejně jako Hra života je Wire World dvojrozměrný externě-součtový automat, je ale čtyřstavový ($k = 4$). Stavy jsou myšleny jako prázdný prostor, vodič, elektron a ,,zadní část" elektronu. Poslední stav je zaveden proto, že v tomto automatu není implementováno nic takového jako elektrické napětí a elektron by sám o sobě nemohl určit, kterým směrem se má po vodiči pohybovat. Takto se pohybuje směrem od zadní části elektronu.

Obrázek 5: Příklady základních hradel sestrojených v automatu Wire World.

K sestavení libovolného logického obvodu a tedy i počítače vystačí hradla AND, OR a NOT. Proto je teoreticky možné sestavit v tomto celulárním automatu digitální počítač a je tedy univerzální. Teoreticky lze se třemi uvedenými hradly sestavit jakékoliv jiné hradlo, lze však sestrojit i ostatní hradla v mnohem jednodušší formě. Dnes už není problém nalézt na internetu hradla XOR, NAND, paměťové bloky, zásobníky, dokonce i displej. Návod na sestrojení součástek je opět na mnoha specializovaných webových stránkách.

2.7 Langtonova smyčka

Langtonova smyčka je speciální druh umělé inteligence. Byla poprvé zmíněna v [5]. Jednotlivé smyčky jsou vlastně mnohobuněčné struktury, ve kterých koluje ,,genetická informace". Po určité době vytvoří smyčka pomocné rameno, které vytvoří dceřinou smyčku se stejným genetickým materiálem. Po dokončení reprodukce se smyčky oddělí a obě se mohou množit dál. Automat se tedy sebereprodukuje. Sebereprodukci prováděl už Neumannův univerzální konstruktor, který byl ale příliš složitý, protože jeho účel byl výpočetní univerzalita. Langton upustil od podmínky univerzality a soustředil se přímo na sebereprodukci. Tak nalezl tento mnohem jednodušší model. Na obrázku 6 je zobrazen proces reprodukce. První smyčka je mateřská. Po reprodukci se vytvořila druhá smyčka se stejným genetickým materiálem.



Obrázek 6: Langtonova smyčka.

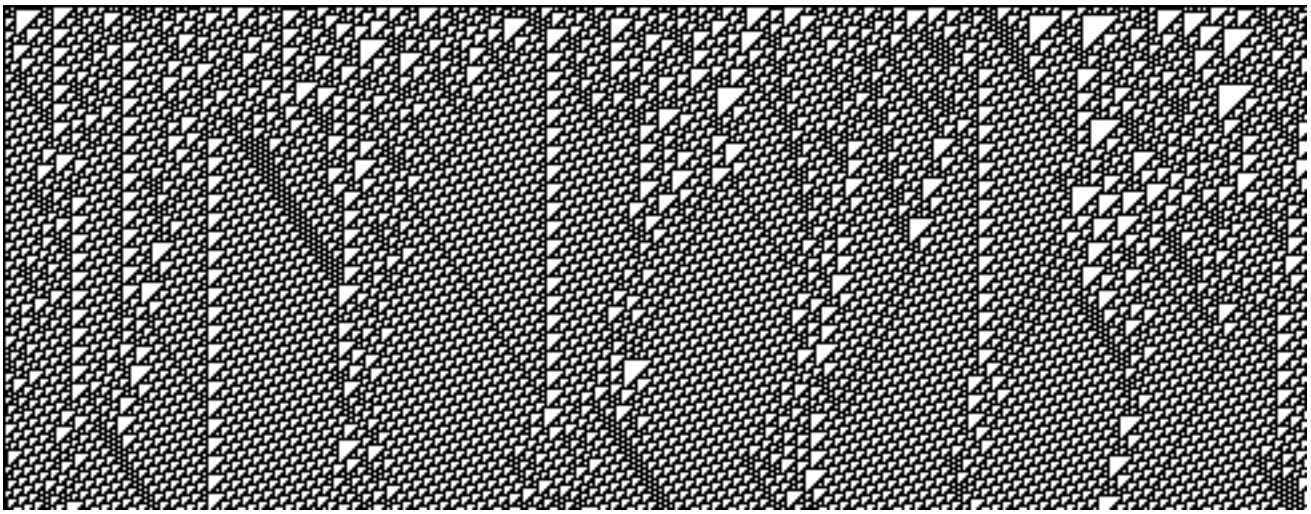
Po vzoru Langtonovy smyčky byly sestrojeny i další automaty. Například Evoloop zahrnuje mutace genetické informace, vytvořená dceřinná buňka tedy může být rozdílná od mateřské. Dále to jsou SDSR smyčky, které mají omezenou životnost, tj. po určité době buňky umírají.

3 Univerzalita pravidla 110

Když si zobrazíme evoluční diagramy všech základních celulárních automatů, pravidlo 110 mezi nimi viditelně vyniká. Všechny ostatní tvoří příliš jednoduché, stabilní konfigurace, nebo naopak chaotickou změť. V pravidle 110 lze však už na nepříliš velkém prostoru pozorovat pohybující se a kolidující struktury, které vyvstanou z počátečního chaosu. Jiná pozoruhodná vlastnost tohoto pravidla je, že evoluční diagram je pokryt různě velkými pravoúhlými trojúhelníky. Pravidlo 110 není ovšem jediný jednorozměrný automat, který tyto vlastnosti vykazuje. Mezi obecnými, součtovými nebo externě-součtovými jich existuje několik. Wolfram je souhrnně označuje automaty třídy IV. Tato mystická třída je na rozdíl od zbylých tří velmi vzácná, poměr počtu všech automatů ku počtu automatů třídy IV bývá v setinách procenta. Stephen Wolfram uvedl domněnku, že jsou tyto automaty výpočetně univerzální, tedy že jsou schopny provést jakýkoliv algoritmus. Jak vůbec může tak jednoduchý systém jako je elementární celulární automat provádět nějaký výpočet? Základní myšlenka spočívá v tom, že program, tedy jeho algoritmus a data, zakódujeme speciálním způsobem do počáteční konfigurace univerzálního celulárních automatu a automat spustíme. Po určitém počtu časových kroků dospěje automat k výsledku, který musí být opět určitým způsobem zakódovány do jeho konfigurace. Dá se říct, že konfigurace automatu je software a pravidlo automatu je hardware. Jak lze vytušit, k počítání slouží ony pohybující se struktury. Tento jev se nazývá počítání založeno na kolizích (*collision-based computing*). Struktury se totiž mohou srážet a vytvářet nové struktury. Na této myšlence je založeno počítání ve většině jednoduchých celulárních automatech. Tuto vlastnost má samozřejmě i pravidlo 110. V této kapitole si objasníme důkaz, který na počátku devadesátých let provedl Matthew Cook. Protože patří pravidlo 110 do třídy základních celulárních automatů, je to nejjednodušší univerzální celulární automat a možná také jeden z nejjednodušších mechanicky realizovatelných systémů vůbec. V praxi není až tak zajímavé, že pravidlo 110 může provádět sčítání, nebo aproximovat π, ale spíš ona ekvivalence s Turingovým strojem, nebo ostatními výpočetními modely. Slavný Problém zastavení například tvrdí, že nemůže existovat algoritmus, který by určil, zda se daný program zastaví, nebo ne. Tento problém je obdobný i v pravidle 110 – nemůžeme sestrojit program, který by dokázal určit, zda se daná konfigurace stabilizuje. Neexistuje žádná zkratka – jediný způsob, jak to zjistit, je pravidlo 110 spustit a počkat. V této kapitole prezentuji fungující simulaci cyklického tagového systému, která může být zkoumána programem přiloženým k této práci.

3.1 Objekty

V časoprostorovém diagramu na obrázku 7 můžeme pozorovat několik objektů, které se pohybují určitým směrem. Pokud necháme pracovat pravidlo 110 dostatečně dlouho z náhodných počátečních podmínek, vytvoří se periodické pozadí, ve kterém se mohou pohybovat struktury zvané glidery (terminologie převzata z Game Of Life). Glidery se mohou pohybovat zleva, zprava nebo se nepohybovat vůbec. Nejdůležitějším jevem jsou kolize. Ze srážky dvou gliderů může buď vzniknout nový glider, jeden z původních zaniknout,



Obrázek 7: Vývoj pravidla 110 z náhodných počátečních podmínek

nebo mohou zaniknout oba. Na tomto jednoduchém principu je postavena celá myšlenka počítání založeného na kolizích. Zajímavé je, že z náhodné konfigurace, jak je zobrazeno na obrázku 7, budou glidery stále kolidovat i po několika tisících dalších časových krocích. Stejně jako v *Game Of Life* vyvstala i v pravidle 110 potřeba analyzovat a katalogizovat objekty, vyskytující se v konfiguracích tohoto automatu. První komplexnější systém pojmenování zavedl Cook v [14], na což navázali Martínez a McIntosh v [12]. V této práci navrhli regulární jazyk nazvaný $phases f_i\_1$, ve kterém lze intuitivním způsobem vyjádřit jakoukoliv počáteční konfiguraci.

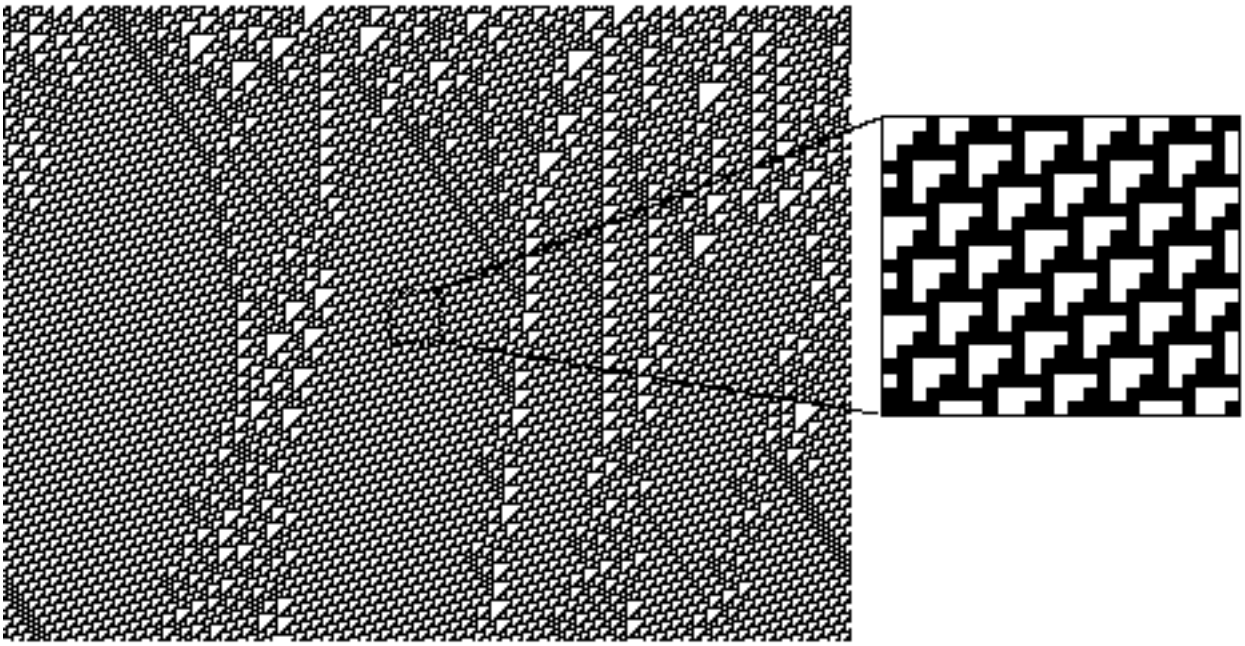
3.1.1 Éter

Éter je periodické pozadí, které vytváří prostředí pro pohyb gliderů. Je to sám o sobě velmi zvláštní typ dlaždice, který se příliš nehodí jako výplň obdélníkového prostoru. Jako vše v evolučním diagramu pravidla 110 je éter složen z trojúhelníků[3]. Jak ukazuje obrázek 8, při náhodné počáteční konfiguraci se éter objeví a vytvoří tak přirozené prostředí pro pohyb gliderů. Téměř ve všech ostatních automatech, které obsahují glidery, vykonává funkci éteru prázdné prostředí, tj. buňky ve stavu 0 (např. v Game Of Life, v součtovém $d = 1, r = 3, k = 2$ automatu 88).

3.1.2 Glidery

Existují dva typy gliderů: základní a rozšířené. Některé glidery mohou být libovolně rozšířené číslem $n \in Z^+$. Z toho plyne, že je možné sestrojit neomezené množství gliderů a navodit tak neomezeně mnoho kolizí. Rozdělením gliderů podle směru pohybu získáme tři skupiny:

[3]Harold V. McIntosh nedávno začal studovat pravidlo 110 jako problém pokrytí plochy různě velkými pravoúhlými trojúhelníky.



Obrázek 8: Periodické prostředí nazvané éter

glidery A^n, D_1 a D_2, které se pohybují zleva. $B, \bar{B}^n, \hat{B}^n, E, \bar{E}^n, F, G^n, H, gun^n$, které se pohybují zprava, nepohyblivé glidery jsou C_1, C_2, C_3. Základní vlastnosti gliderů je rychlost pohybu a jejich perioda, tedy počet časových kroků, po kterých bude jejich konfigurace opět stejná. Rozšiřováním se ani jedna vlastnost nemění. Obrázek 9 ukazuje všechny základní formy gliderů. Éter je na obrázku zašedlý, aby glidery vynikly.

3.2 Popis důkazu

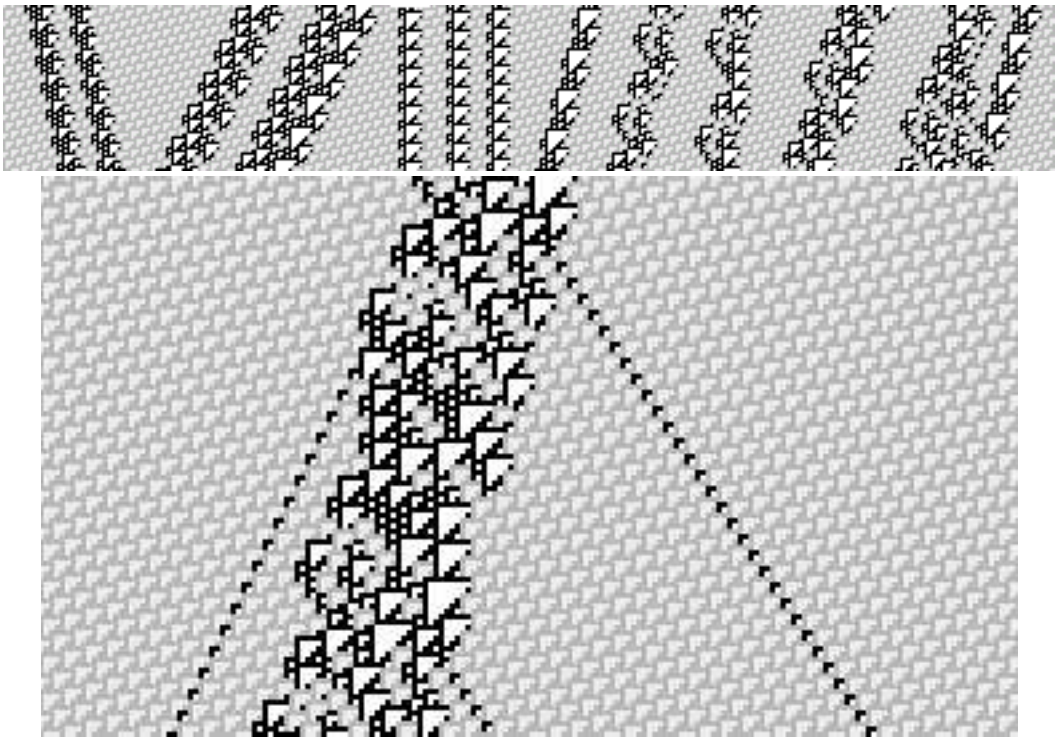
Po von Neumannově univerzálním konstruktoru, který fungoval v dvojrozměrném prostoru s 29 stavy, se objevily i další a jednodušší automaty umožňující univerzální počítání. Viz Codd [15] v 1968, Smith [16] v 1971, Conway [3] v 1974, Lindgren & Nordahl [17] v 1990. Univerzalita pravidla 110 je poslední redukce – byl nalezen nejjednodušší univerzální celulární automat. Důkaz spočívá v simulaci tagového systému v pravidle 110.

3.2.1 Tagový systém

Tagový systém je výpočetní model, který navrhl v roce 1921 jako alternativu Turingově stroji Emil Leon Post a jeho univerzalitu dokázal v roce 1961 Marvin Minsky v [9]. Je to druh substitučního systému, jinými slovy funguje na principu nahrazování slov. Prakticky je to stroj, který má jednosměrně nekonečnou pásku, a operace, které zvládá, jsou čtení prvního znaku na pásce, mazání prvních několika znaků a přidávání znaků na konec pásky.

Definice

m-tagový systém $\mathcal{T}$ je čtveřice (Σ, m, A, P), kde



Obrázek 9: Systém gliderů. $D_1, D_2, \bar{B}, \hat{B}, C_1, C_2, C_3, E, \bar{E}, F, G, H, Gun$ produkující A a B

- $\Sigma = \{a_0, a_1, ..., a_{n-1}\}$ je abeceda obsahující n symbolů, z nichž jeden může být speciální zastavující symbol.
- $m \in N$ je takzvané mazací číslo.
- A je počáteční slovo nad abecedou Σ.
- P je množina pravidel, přiřazující každému symbolu jedno slovo:
  $a_0 \rightarrow a_{0,1}a_{0,2}\ldots a_{0,l_0}$
  ...
  $a_{n-1} \rightarrow a_{n-1,1}a_{n-1,2}\ldots a_{n-1,l_{n-1}}$
  kde každé $a_{i,j} \in \Sigma, 0 \leq i < n$.

Případnému zastavujícímu symbolu přiřazujeme opět stejný zastavující symbol. Na začátku výpočtu obsahuje páska stroje slovo A. V prvním kroku systém přečte první symbol na pásce, podle pravidel zjistí odpovídající slovo a to zapíše na konec pásky, neboli zřetězí toto slovo s původním slovem A. Následně pak smaže prvních m políček z pásky a tím přejde do druhého kroku výpočtu. Systém se zastaví, pokud je délka slova menší než m, nebo pokud je první symbol zastavující.
Příklad jednoduchého 2-tagového systému: $\Sigma = \{a, b, c, H\}$
Pravidla:

$a \rightarrow ccbaH$
$b \rightarrow cca$
$c \rightarrow cc$
$H \rightarrow H$ (zastavující symbol)
$A = baa$ (začínající slovo)
Výpis práce: `baa acaa caccbaH ccbaHcc baHcccc Hcccccca zastavení`

3.2.2 Cyklický tagový systém

Marvin Minsky dokázal, že pro každé $m > 1$ platí, že pro každý Turingův stroj existuje m-tagový systém, který tento stroj simuluje. Právě v m-tagových systémech viděl Cook možnost, jak dokázat univerzalitu pravidla 110. Vytvořil nový model, cyklický tagový systém (CTS), a dokázal jeho ekvivalenci s m-tagovými systémy a s pravidlem 110. Tím vytvořil most, který tyto dva modely spojil a dokázal tak univerzalitu pravidla 110. Ukážeme si, jak může být jakýkoliv m-tagový systém simulován cyklickým tagovým systémem, neboť je to velmi důležitá část důkazu. Cyklický tagový systém je svou konstrukcí velmi podobný tagovým systémům. Obsahuje však pouze dva symboly $\Sigma = \{0, 1\}$, mazací číslo je vždy $m = 1$ a pravidla se uplatňují jiným způsobem. Pravidlo je množina tzv. produkčních slov, která se cyklicky uplatňují pro iterace systému. Pokud je první znak slova symbol 1, přidá se aktuální produkční slovo pro tento krok na konec slova. První symbol se pak smaže. Systém se zastaví pouze, když je délka slova nulová. Převod m-tagového systému do CTS je poměrně snadno pochopitelný. Jako příklad mějme 2-tagový systém:
$\Sigma = \{a, b, H\}$
Pravidla:
$a \rightarrow ab$
$b \rightarrow aaH$
$H \rightarrow H$
$A = ba$
Výpis práce: `ba aaH Hab zastavení`

Pro emulaci cyklickým tagovým systémem každý symbol ze Σ nahradíme unikátním slovem nad $\{0, 1\}$ s délkou n (mohutnost množiny Σ):
$a = 100$
$b = 010$
$H = 001$
S použitím této náhrady sestrojíme taky produkční slova:
$ab = 100010$
$aaH = 100100001$
$H = 001$
Celkový počet produkčních slov musí být mn, proto ji doplníme prázdnými slovy ϵ. Množina cyklických pravidel je tedy $\{100010, 100100001, 001, \epsilon, \epsilon, \epsilon\}$
Zakódujeme ještě počáteční slovo $ba = 010100$ a můžeme spustit systém:

krok	pravidlo	páska
0	100010	$010100 = ba$
1	100100001	10100
2	001	0100100100001
3	ϵ	100100100001
4	ϵ	00100100001
5	ϵ	0100100001
6	100010	$100100001 = aaH$
7	100100001	00100001100010
8	001	0100001100010
9	ϵ	100001100010
10	ϵ	00001100010
11	ϵ	0001100010
12	100010	$001100010 = Hab$ (myšlené zastavení)
13	100100001	01100010
14	001	1100010
⋮	⋮	⋮

Každý krok dělitelný číslem mn reprezentuje jeden krok simulovaného m-tagového systému. CTS se ovšem nezastaví na zastavujícím symbolu, ale jen když bude délka slova na pásce nulová, což mimochodem v tomto případě nikdy nenastane. Proto je zastavení jen myšlené. Převedli jsme tedy tagový systém do cyklického tagového systému.

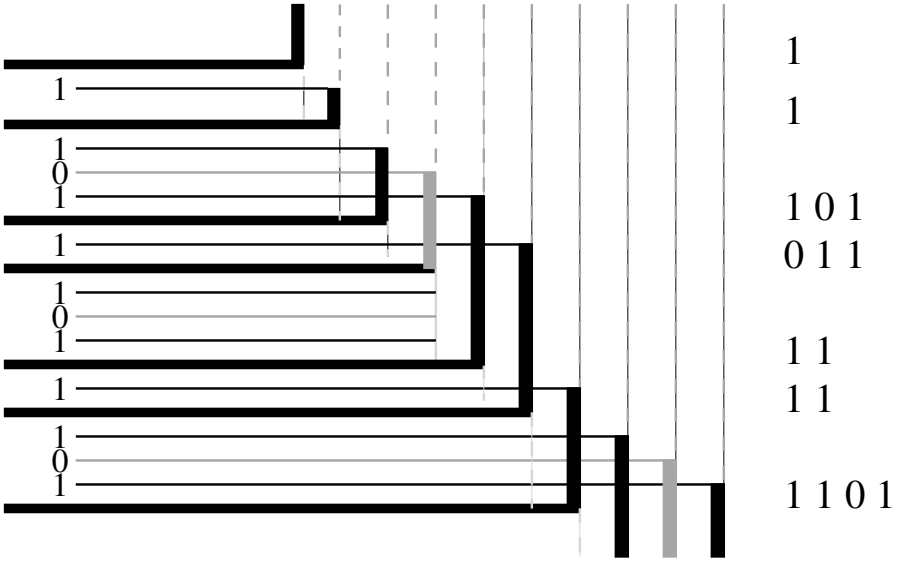
3.3 Simulace cyklického tagového systému pravidlem 110

V této kapitole ukážeme, jak může být práce cyklického tagového systému provedena základním celulárním automatem 110. CTS, který byl předveden v předchozí sekci, je už poměrně složitý. Systém, na kterém provedeme redukci na pravidlo 110, je mnohem jednodušší – má pouze dvě produkční slova, která jsou navíc velmi krátká. Důležité ale je, že princip simulace je stejný bez závislosti na složitosti simulovaného systému.
Definujme si cyklický tagový systém s produkčními slovy $\{1, 101\}$ s počátečním slovem '1':

pravidlo	páska
1	1
101	1
1	101
101	011
1	11
101	11
1	1101
101	1011
1	011101
⋮	⋮

Pochopitelně budou veškeré funkce cyklického tagového systému (čtení, mazání, přidávání)

realizovány v pravidle 110 pomocí gliderů a jejich kolizí. Na obrázku 10 je zobrazen zjednodušený diagram simulace. Tenké linky pohybující se zleva představují *symboly produčních slov*. Tlustší linka mířící zleva je tzv. *oddělovač*. Tenké linky vedené odshora jsou *spouštěče*, tlusté linky jsou *symboly* na pásce. Mechanismus funguje tak, že se oddělovač pohybuje, dokud nenarazí na první symbol na pásce. Pokud je to 1, vytvoří oddělovač *filtr*, který změní symboly produkčního slova na speciální *přidávací signály*. Pokud je symbol 0, vytvoří oddělovač blok, který symboly produkčního nepropustí. Jednotlivé propuštěné přidávací signály putují dál, dokud je spouštěč nepřemění na symbol 0 nebo 1.



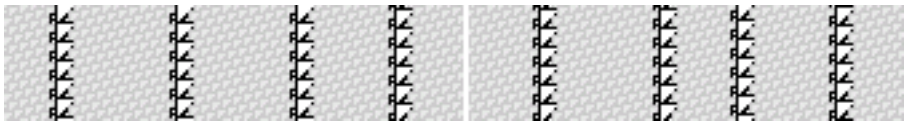
Obrázek 10: Diagram simulace CTS v pravidle 110

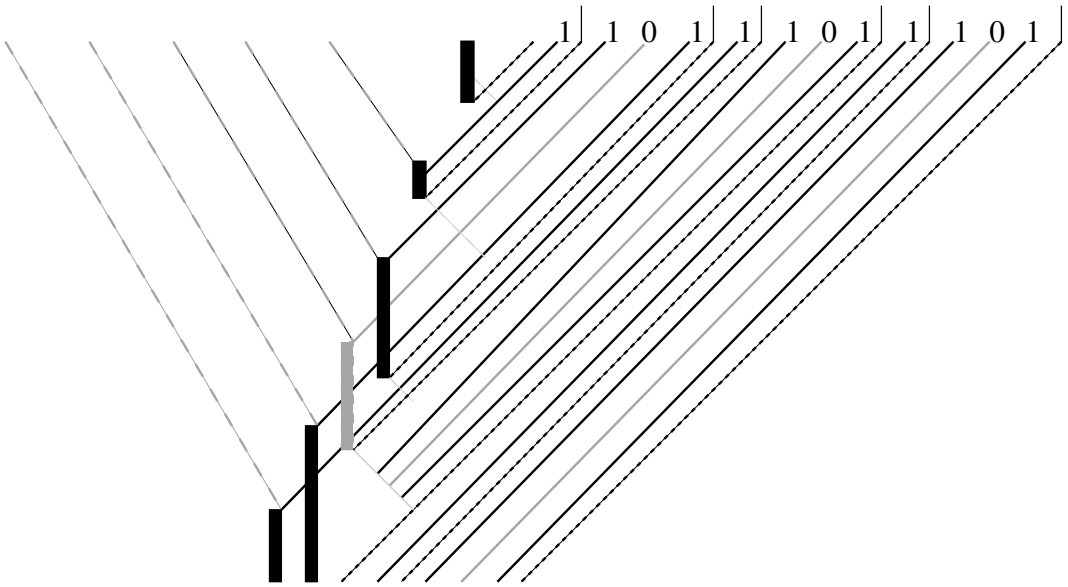
Obrázek 10 je zkonstruován tak, aby byla činnost jednoduše pochopitelná a viditelně zřejmá, ve skutečnosti však simulace vypadá jinak. Reálnější zobrazení simulace můžeme vidět na obrázku 11. Princip však zůstává stejný.

3.3.1 Signály

Nyní si ukážeme, co vlastně ony signály zastupují ve skutečnosti. Bez výjimky to nejsou pouhé glidery, ale skupiny gliderů.

Signály 0 a 1
Signály 1 a 0 vyjadřují symboly na pásce CTS. Jsou složeny ze čtyř C_2 gliderů. Rozdíl mezi symboly závisí na velikosti mezer mezi jednotlivými glidery. Na obrázku jsou signály 1 a 0 v tomto pořadí.

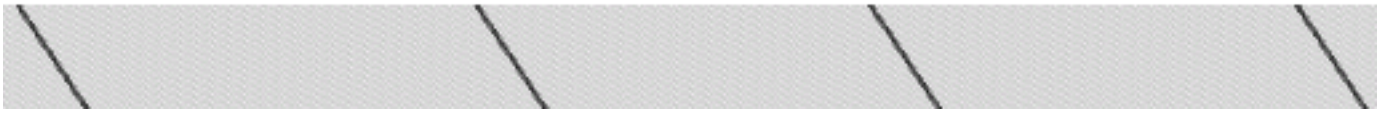




Obrázek 11: Věrnější diagram simulace CTS v pravidle 110

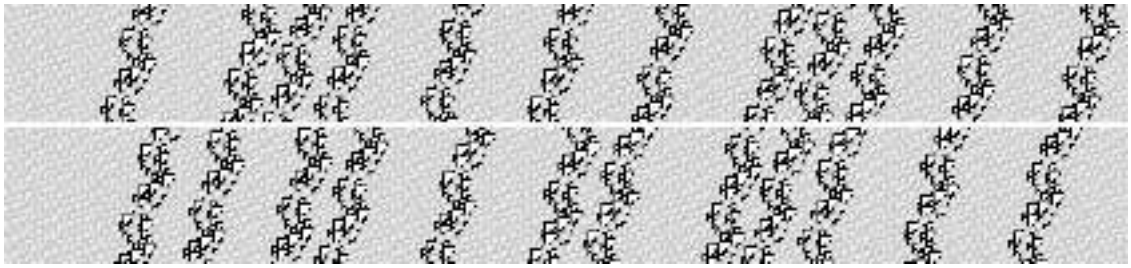
Spouštěč

Funkce spouštěče je schopnost přeměnit přidávací signál 0 nebo 1 na symbol 0 nebo 1. Spouštěč je složen ze čtyř A^4 gliderů. Protože je A nejrychleji se pohybující glider, musí být spouštěče umístěny do velké vzdálenosti. Tato levá část zabírá největší procento z celkové velikosti počáteční konfigurace.



Symboly producních slov

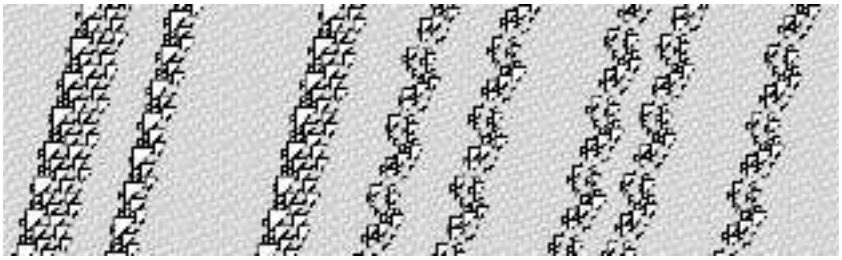
Symboly produkčních slov jsou nejsložitější signály, skládají se ze dvanácti gliderů. Zároveň je to také nejfrekventovanější signál v simulaci. To je další faktor způsobující velký rozměr počáteční konfigurace. Obrázek zobrazuje signál 1 a 0.



Oddělovač

Oddělovač je poslední struktura, která se vyskytuje přímo v počáteční konfiguraci. Ostatní

signály, jako přidávací symboly nebo filtry, vzniknou až v průběhu simulace. Oddělovač reaguje rozdílně, když se srazí se symbolem 0 nebo symbolem 1. Podle toho vytvoří filtr, který buď propouští symboly produkčních slov, nebo ne. Je zobrazen na následujícím obrázku.



3.3.2 Průběh simulace

Nejdůležitější fáze simulace mohou být zobrazeny přes aplikaci, která je přiložena k této práci (viz příloha A). Simulace s názvem *smazani_symbolu_1* zobrazuje první krok, tedy smazání prvního symbolu na pásce. Simulace *vytvoreni_filtru* je přímé pokračování předchozí simulace. Protože byl smazaný symbol 1, vytvoří se filtr, který symboly produkčních slov přemění na přidávací signály. Simulace *vytvoreni_bloku* zobrazuje vytvoření bloku, který vznikne po smazání symbolu 0. Konečně simulace *vytvoreni_symbolu_1* zobrazuje setkání spouštěče s přidávacím symbolem 1, čímž vzniká symbol 1 na pásce.

4 Závěr

Záměrem práce bylo popsat problematiku celulárních automatů a vysvětlit, jak mohou ty nejjednodušší z nich provádět výpočty. Konkrétní cíl byl sestavit fungující simulaci cyklického tagového systému v základním celulárním automatu 110. Čtyři kroky simulovaného systému jsou obsaženy v 3 782 602 600 buňkách (65 899 buněk v 57 400 časových krocích). Počítání v jednoduchých celulárních automatech je tedy značně neefektivní a lze si jen těžko představit, že by bylo využíváno v praxi. Neznamená to ale, že by byly celulární automaty zbytečné. Jedna ze základních tezí teoretické informatiky tvrdí, že čím je mechanismus jednodušší, tím obtížnější je práce s ním. V terminologii celulárních automatů to znamená, že čím je počet stavů buněk menší a přechodová funkce jednodušší, tím větší množství buněk je potřeba ke stejnému výpočtu. V dnešní době se například začínají rozmáhat víceprocesorové počítače. Dalo by se říct, že se jedná o celulární automat, jehož buňky jsou jednotlivé procesory. Další možné budoucí využití principů celulárních automatů je v sestavování a programování kvantových počítačů. Složitější celulární automaty jsou tedy ve středu zájmu, je však také důležité pochopit jednoduché celulární automaty, ze kterých všechny myšlenky vychází. Zajímavé, že tak jednoduché mechanismy, jakými základní celulární automaty bezesporu jsou, mohou z teoretického hlediska simulovat jakýkoliv jiný počítací stroj. Jsou tedy univerzální. Otevřenou otázkou zůstává, jak jednoduché mohou univerzální stroje být. Tento limit doposud vytváří celulární automat 110.

5 Poděkování

Poděkování patří lidem z diskusní skupiny comp.theory.cell-auto, kteří mi mnohokrát pomohli s mými otázkami. Dále chci poděkovat Mirkovi Rahnemu a Genaro Martínezovi, kteří mi pomohli pochopit univerzalitu pravidla 110 svými pracemi i osobními radami. Poděkovat chci také mému bratrovi Adamovi Šiškovi, jehož připomínky a dotazy mi pomáhali tříbit mé argumenty.

A Příloha

K této práci je přiložen program na zkoumání základních celulárních automatů, zejména pravidla 110. Je to soustava programu v jazyce C a GTK grafického prostředí napsaného v Pythonu. Tento program byl vytvořen přímo pro potřeby prezentace této práce. Pro zkoumání celulárních automatů doporučuji používat programy MCell (*http://www.mirekw.com*), CelLab (*http://www.fourmilab.ch/cellab/*) nebo jiné.

Reference

[1] *Ivana Černá, Mojmír Křetínský, Antonín Kučera* **Automaty a formální jazyky I**, 2002, Fakulta Informatiky MUNI

[2] *Stephen Wolfram* **Cellular Automata as Simple Self-Organizing Systems**, 1982, `http://www.stephenwolfram.com/publications/articles/ca/82-cellular`

[3] *R. T. Wainwright* **Life Is Universal!**, 1974

[4] *Paul Rendel* **A Turing Machine In Conway's Game Of Life**, 2001, `http://rendell-attic.org/gol`

[5] *Christopher G. Langton* **Self-Reproducing in Cellular Automata**, 1984

[6] *Christopher Langton* **Computation at the edge of chaos: phase transitions and emergent computation**, 1990

[7] *Stephen Wolfram* **Cellular Automata**, 1983, `http://www.stephenwolfram.com/publications/articles/ca/83-cellular`

[8] *Stephen Wolfram* **A New Kind Of Science**, 2002, `http://www.nksonline.com/`

[9] *Marvin Minsky* **Recursive unsolvability of Post's problem of tag and other topics in the theory of Turing machines**, 1961, Annals of Mathematics 74, 437-455.

[10] *Edward Fredkin, T. Toffoli* **Conservative Logic**, 1982, Int. Jour. of Theo. Phy. 21, 219.

[11] *Melanie Mitchell, James Crutchfield, Peter Hraber* **Evolving Cellular Automata to Perform Computations: Mechanisms and Impediments**, 1993, `http://www.santafe.edu/ evca/index.html`

[12] *Genaro Martínez, H. McIntosh, J.S. Mora, S.C. Vergara* **Determining a regular language by glider-based structures called phases fi1 in Rule 110**, 2006, `http://uncomp.uwe.ac.uk/genaro/papers.html`

[13] *Genaro Martínez, H. McIntosh, J.S. Mora, S.C. Vergara* **Reproducing the cyclic tag system developed by Matthew Cook with Rule 110 using the phases fi_1**, v rozpracováni, posláno osobně.

[14] *Mattew Cook* **Universality in Elementary Cellular Automata**, 2004, Complex Systems 15 (1), 1-40

[15] *E. F. Codd* **Cellular Automata**, 1968, Academic Press, Inc. New York and London.

[16] *A. R. Smith III*, **Simple computation-universal cellular spaces**, 1971, J. of the Assoc. for Computing Machinery 18, 339353.

[17] *K. Lindgren, M. G. Nordahl*, **Universal Computation in Simple One-Dimensional Cellular Automata**, 1990, Complex Systems 4, 229318.

[18] *Andrew Ilachinski*, **Cellular Automata: A Discrete Universe**, 2001, World Scientific Publishing