VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

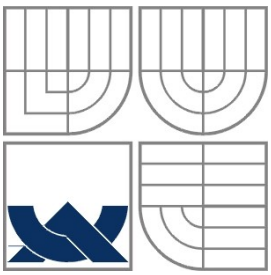
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

SEBEREPLIKACE V CELULÁRNÍCH AUTOMATECH

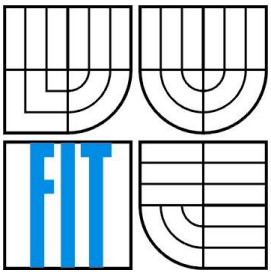
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MARTIN MIKEŠ
AUTHOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

SEBEREPLIKACE V CELULÁRNÍCH AUTOMATECH

SELF-REPLICATION IN CELLULAR AUTOMATA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
MARTIN MIKEŠ

VEDOUCÍ PRÁCE
SUPERVISOR
ING. LUDĚK ŽALOUDEK

BRNO 2010

Abstrakt

Tato práce představuje celulární automaty jako systémy, které mohou sloužit jako prototypový model pro komplexní masivně paralelní systémy složené z jednoduchých, navzájem interagujících komponent. Zaměřuje se na sebereplikující se smyčky – struktury operující v celulárním prostoru, mající potenciálně možnost provádět užitečné úlohy. Je zde ukázána implementace tří takových smyček, ve kterých jsou demonstrovány úlohy jednobarevné vyplnění vnitřní plochy smyčky, konstrukce písmen „MM“ a binární sčítání. Na závěr je diskutována praktická použitelnost těchto principů.

Abstract

This thesis introduces cellular automata as an environment suitable for simulating complex and massively parallel systems, built from a large number of simple cooperating units. The thesis explores self-replication and self-replicating loops – structures operating within the cellular space, particularly those capable of carrying out useful tasks. The thesis provides a description of implementation of three such loops, executing following tasks: coloring of the space within a loop, construction of letters “MM” and binary addition. A practical usability is discussed at the end.

Klíčová slova

Celulární automaty, sebereplikace, emergence, sebereplikující se smyčky.

Keywords

Cellular automata, self-replication, emergence, self-replicating loops.

Citace

Martin Mikeš: Sebereplikace v celulárních automatech, bakalářská práce, Brno, FIT VUT v Brně, 2010

Sebereplikace v celulárních automatech

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením ing. Luďka Žaloudka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Mikeš
19. 5. 2010

Poděkování

Chtěl bych poděkovat svému vedoucímu ing. Luďkovi Žaloudkovi za jeho přínosné připomínky a nápady a jeho pečlivý přístup k vedení mé práce, kterou mi tím tolik ulehčil. Dále bych rád poděkoval svým rodičům za podporu během studia.

Obsah

1 Úvod

Jedním z trendů ve vývoji výpočetních systémů je využívání systémů složených z velkého množství jednoduchých paralelně pracujících jednotek. Při dostatečné jednoduchosti takové jednotky se nabízí možnost stvořit výsledný systém levně, výpočet založený na paralelismu může být příslibem vysokého výkonu systému. Vhodným teoretickým modelem pro zmíněné systémy jsou celulární automaty.

Celulární automaty (CA) jsou dynamické systémy, které mohou sloužit jako prototypový model pro komplexní systémy a procesy využívající velké množství identických, jednoduchých, navzájem interagujících komponent. Lze v nich studovat náznaky chování, kdy jednoduché prvky kooperují a zdánlivě se samy organizují obdobně jako v reálných systémech. V padesátých letech minulého století si je zvolil John von Neumann jako platformu pro svůj výzkum, ve kterém chtěl vyvinout stroj schopný vytvořit kopii sama sebe. Vytvořil takzvaný univerzální konstruktor, automat schopný výpočetní a konstrukční univerzality, ten je ale do dnešní doby příliš složitý na simulaci na současných počítačích. Výzkum se vydal cestou zjednodušení, vznikla Langtonova smyčka – jednoduchý automat schopný sebereplikace. V dalším vývoji pak byla vyvíjena snaha obdařit jednoduché smyčky schopností provádět užitečné úlohy. Cílem práce je popsat modely těchto smyček a demonstrovat na nich proveditelné úlohy, z čehož by mělo vyplynout, jaké typy úloh jsou realizovatelné. S ohledem na praktické využití takových smyček v reálných systémech je požadavkem, aby smyčky efektivně vyplnily prostor celulárního automatu svými replikami, čímž by se do prostoru distribuovaly jednotky provádějící implementované úlohy.

Ve druhé kapitole této práce nejprve představuji celulární automaty. Jde o teoretický model pracující s diskrétním prostorem a časem. Jeho podstatou je n-rozměrná mřížka buněk definovaných svým stavem, který se v časových krocích mění jako funkce stavů dané buňky a buněk v blízkém okolí. Diskutovány jsou také možnosti využití jednodimenzionálních (1D), dvoudimenzionálních (2D) a třídimenzionálních (3D) celulárních automatů.

Třetí kapitola popisuje celulární automaty jako prostředí vhodné pro výzkum emergentních jevů – chování známého z reálného světa, objevujícího se prostředí interagujících elementů bez explicitního řízení na úrovni celého systému.

Kapitola čtvrtá popisuje sebereplikaci a sebereplikující se smyčky jako jeden z emergentních jevů. Celulární automaty, ve kterých se sebereplikace objevuje, jsou však výsledkem cílené snahy navrhnout množinu pravidel řídící daný automat. Nejznámějšími takovými automaty jsou von Neumannův univerzální konstruktor a Langtonova smyčka.

Kapitola pátá představuje dvě smyčky schopné provádět užitečné výpočty: Tempestiho a Perrierovu smyčku. Je zde popsán princip jejich chování a jejich schopnosti. Na závěr je zhodnocena vhodnost smyček pro cíle této práce.

V šesté kapitole popisuji implementaci zvolených úloh jako spustitelných programů prováděných v Tempestiho smyčce.

Závěrečná sedmá kapitola přináší zhodnocení mé práce s celulárními automaty a vyhodnocení celulárních automatů jako platformy pro výzkum podobných systému založených na lokalitě interakcí a masivně paralelním zpracování.

2 Celulární automaty

Jako platformu pro svůj výzkum poprvé použil celulární automat ve čtyřicátých letech minulého století John von Neumann poté, co se nechal inspirovat mřížkovým modelem (lattice model), který používal pro výzkum růstu krystalů Stanisław Ulam.

Celulární automat je deterministický matematický model pracující s diskrétním časem, který je charakteristický interakcemi pouze v malém okolí a jejich paralelním průběhem. Celulární automat je tvořen polem prvků – buněk, uspořádaných do n-rozměrné mřížky. Každá buňka celulárního automatu pracuje jako konečný stavový automat; ty mohou být všechny identické, potom hovoříme o uniformních celulárních automatech, nebo jich mohou být dva a více typů v neuniformních celulárních automatech. Každá buňka je v daném časovém okamžiku charakterizována jedním z konečné množiny stavů, z nichž jeden bývá často označen jako neaktivní stav, vyplňující celulární prostor. Do celulárního prostoru je pak umístěno uspořádání buněk v aktivních stavech tvořící takzvanou počáteční konfiguraci. V každém časovém okamžiku je na základě pravidel celulárního automatu synchronně pro všechny buňky vypočítán jejich nový stav. Pravidla sestávají ze tří částí, popisující současný stav buňky, její blízké okolí a stav buňky v následujícím časovém okamžiku. Pravidly se tak řídí chování celulárního automatu[1].

2.1 Jednorozměrné celulární automaty

Nejjednodušší celulární automaty jsou jednorozměrné. Definovány mohou být pomocí sedmice [2]:

$$\mathbf{A} = (Q, N, R, z, b_1, b_2, c0)$$

kde

Q je binární množina stavů,
N označuje používané okolí buňky,
z vyjadřuje počet buněk,
b_1 a b_2 vymezují hraniční hodnoty,
c_0 je počáteční konfigurace

a mapováním $R : S \rightarrow (Q^N \rightarrow Q)$,
které přiřazuje každé buňce $S = \{1, 2, \ldots, z\}$ v mřížce lokální přechodovou funkci $\delta_1, \delta_2 \ldots, \delta_z$ kde $\delta i : Q^N \rightarrow Q$, $i \in S$.
Konfigurací **A** je potom mapování $c \in Q^S$, které přiřazuje stav každé buňce. Pokud se za používané okolí bere $N = \{-1, 0, 1\}$, neboli buňka a její dva bezprostřední sousedé, globální přechodová funkce G: $Q^S \rightarrow Q^S$ je definovaná takto:

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

kde c_i označuje konfiguraci celulárního automatu v kroku i.

Prostor automatu je v ideálním případě nekonečně velký, při implementaci se však musí zavést omezení a je proto třeba pro buňky na okrajích celulárního automatu definovat tzv. okrajové podmínky, aby všechny buňky měly dva sousedy. Nejčastější řešení jsou konstantní okrajové podmínky, kdy se buňkám na okrajích pevně nastaví hodnoty sousedů, nebo cyklické okrajové podmínky, kdy se ustanoví, že okrajové buňky spolu sousedí, čímž se z prostoru automatu de facto vytváří kruh, v případě dvoudimenzionálního automatu pak toroid.

Pro buňku a její okolí existuje 2^3=8 kombinací hodnot, které tyto 3 buňky mohou nabývat, pro řízení 1D celulárního automatu je tak potřeba 8 pravidel. Stavový automat potom na základě stavu buňky, jejího okolí a příslušného pravidla určí stav buňky v následujícím kroku. Existuje proto 2^8=256 jednoduchých celulárních automatů. Popisovány jsou Wolframovým kódem, tj. číslem od 1 do 255, automat se pak označuje <pravidlo+Wolframův kód>. Binární reprezentace Wolframova kódu určuje výsledné stavy buněk pro všechny kombinace hodnot. V tabulce je uveden příklad pravidel pro celulární automat s Wolframovým kódem 30 (binárně 00011110).

Stávající konfigurace	111	110	101	100	011	010	001	000
Nový stav buňky	0	0	0	1	1	1	1	0

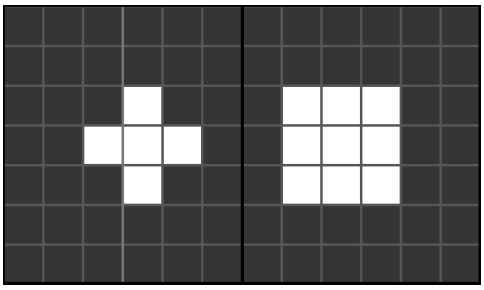
Tab. 2.1 Formát pravidel 1D celulárních automatů.

Jednodimenzionální celulární automaty často nabízejí zajímavé využití, jako například generování pseudonáhodných čísel a šifrování (pravidlo 30), simulace usazování částic na površích, modelování dopravního provozu či vyřešení problému většiny (pravidlo 184) [3].

2.2 Dvourozměrné celulární automaty

Množina stavů dvourozměrných automatů obvykle obsahuje 2 až několik desítek stavů. Používaná okolí existují dvě: Pro čtvercovou buňku to mohou být čtyři buňky, které mají s danou buňkou společnou stranu; takové okolí se nazývá von Neumannovo. Okolí osmi buněk, které mají s danou buňkou společnou stranu nebo vrchol, se nazývá Moorovo. Obě jsou znázorněna na obrázku 2.1. Pro každou buňku existuje n^q pravidel, kde n je počet buněk v uvažovaném okolí (5 pro von Neumannovo, 9 pro Moorovo) a q je počet stavů, se kterými automat pracuje.

Dvourozměrné celulární automaty lze využit v širokém spektru aplikací. Lze pomocí nich zkoumat formaci vzorů v obrazu a generovat je či zkoumat umělý život a evoluci „organismů“ obývajících prostor celulárního automatu, ale jejich hlavní doménou jsou simulace: Lze simulovat například propagaci požáru, rozrůstání měst nebo změny ve využívání půdy[4]. Tommaso Toffoli navrhl využití celulárních automatů jako alternativy řešení diferenciálních rovnic. V celulárních automatech lze také simulovat libovolný počítačový algoritmus, což dokázal Moshe Sipper, když v neuniformním celulárním automatu pracujícím s von Neumannovým okolím naimplementoval dvouregistrový stroj podle konceptu Marvina Minského, který je ekvivalentní Turingovu stroji [5].



Obr. 2.1 Moorovo okolí (vpravo) a von Neumannovo okolí (vlevo).

2.3 Trojrozměrné celulární automaty

3D celulární automaty používají okolí odpovídající von Neumannovu a Moorovu okolí, tam však mají 7 resp. 27 buněk. Jsou proto vhodná tam, kde lze pravidla generovat automaticky – příklad tohoto jsou simulace, kde pravidla odrážejí podstatu fyzikálních zákonů. 3D CA se tak používají například k simulaci rozptylu chemikálií, chování tekutin a modelování. Uplatnění mají i v analýze 3D snímků, používaných například v medicíně.

2.4 Celulární automaty - shrnutí

Celulární automaty jsou příhodnou platformou pro výzkum a simulace; aplikují se v řadě oblastí jako je fyzika, biologie, chemie, biochemie či geologie. Předmětem zájmu jsou díky jejich třem klíčovým vlastnostem [5]:

1) Umožňují paralelní zpracování: Na počítačích se paralelní zpracování simuluje, ale v prostředí, kde každá buňka disponuje svojí vlastní výpočetní jednotkou, lze implementovat rozsáhlé systémy.

2) Všechny buňky jsou řízeny pouze interakcí se svým bezprostředním okolím. Není zde přítomný žádný prvek řídící automat centrálně, s narůstající komplexností systému se tak nezvyšují nároky na jeho režii.

3) Buňky automatu jsou velmi jednoduché, řízené konečným stavovým automatem.

Díky těmto vlastnostem jsou celulární automaty ideálním prostředím pro studium emergentních jevů.

3 Emergence

Emergence je vynořování složitých jevů a vzorů v systému tvořeném jednoduchými vzájemně interagujícími prvky. Emergentními jevy v jednodimenzionálních celulárních automatech se podrobně zabýval Stephen Wolfram. Podle typu chování je rozdělil do 4 tříd [3].

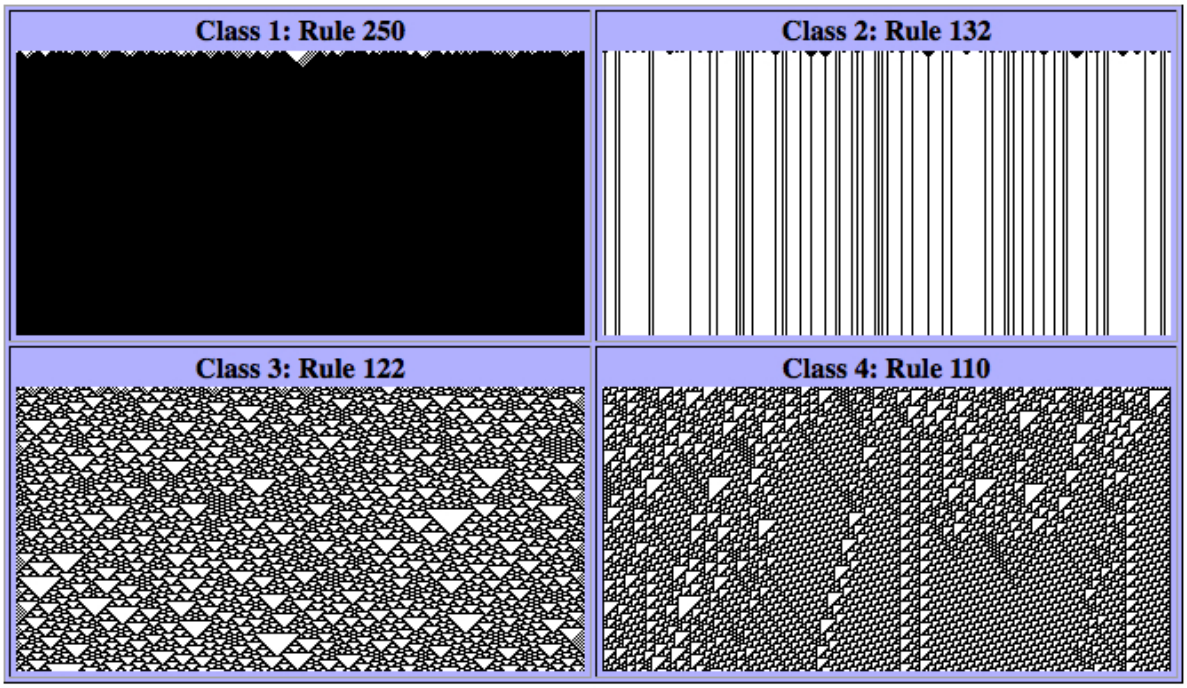
Třída 1: Celulární automaty patřící do třídy 1 vykazují nejjednodušší chování. Z většiny náhodných původních vzoru se rychle vyvine statická sestava, která se již dále nemění.

Třída 2: Vývoj náhodné počáteční konfigurace spěje k jednoduchým stabilním nebo oscilujícím strukturám – ty se obnovují s periodou několika málo kroků.

Třída 3: Většina počátečních konfigurací vede k aperiodickému, chaotickému náhodně vypadajícímu chování. Rozsáhlejší stabilní struktury většinou rychle zanikají, ale často se objevují trojúhelníky nebo jiné drobné útvary.

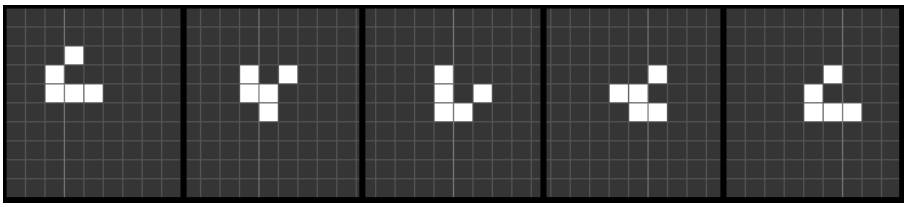
Třída 4: V celulárních automatech 4. třídy se objevuje nejsložitější chování, kombinující pravidelné struktury 2. třídy a chaotickou proměnlivost 3. třídy. Objevují se zde složité vzory, které se pohybují prostorem a interagují spolu. Ve dvoudimenzionálních automatech 4. třídy lze simulovat algoritmy s výpočetní univerzálností.

Ukázky automatů jednotlivých tříd jsou na obrázku 3.1.



Obr. 3.1 Wolframovy třídy. 1D CA č. 250, 132, 122 a 110 znázorňují chování celulárních automatů třídy 1, 2, 3 a 4.

Zajímavých případů emergentního chování ve dvoudimenzionálních celulárních automatech je mnoho. Mezi nejjednodušší jevy patří takzvané kluzáky (na obrázku 3.2) – útvary sestávající z více buněk, které cestují prostorem buněčného automatu – nebo oscilátory – seskupení, která periodicky obnovují svoji strukturu. Lze je pozorovat například v binárním automatu Game of Life [6]. Některé 2D celulární automaty produkují fraktálové obrazce. Hiroki Sayama zkonstruoval celulární automat, ve kterém lze pozorovat evoluci „organismů“, obývajících celulární prostor, které soupeří o místo [7]. Jedním z nejzajímavějších jevů v celulárních automatech je však sebereplikace.



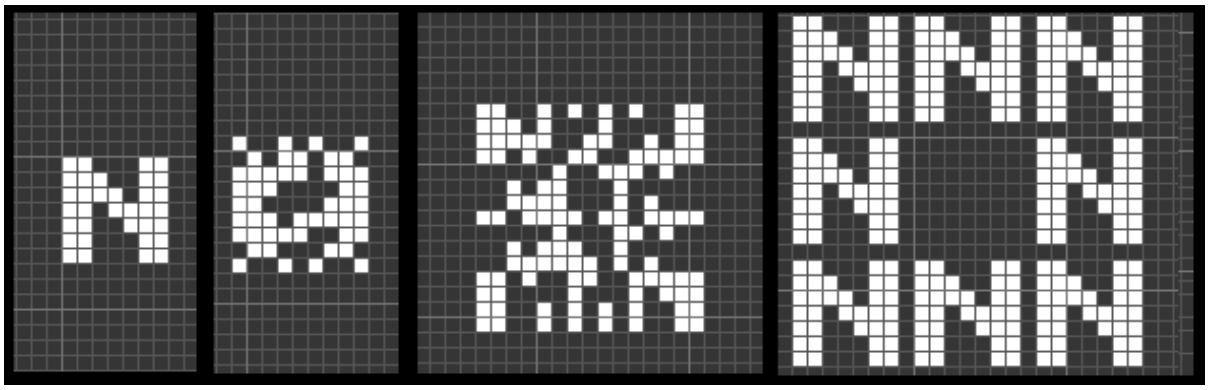
Obr. 3.2 Nejjednodušší kluzák v 2D binárním automatu Game of Life.

4 Sebereplikace

Sebereplikující se struktura je reprezentována konfigurací několika shluknutých buněk, která po několika výpočetních krocích vytvoří izolovaně na jiném místě celulárního prostoru svou repliku, která může být pootočena [8].

4.1 Primitivní replikace

Nejjednodušší sebereplikace v celulárních automatech lze dosáhnout v binárních automatech řízených pravidly parity, kde nový stav buňky je odvislý od počtu buněk ve stavu 1 v jejím okolí. V takovém systému je libovolný vzor schopen replikace, jak je ukázáno na obrázku 4.1.



Obr. 4.1 Replikace pomocí pravidel parity, průběh v krocích 0, 1, 4, 8.

4.2 Univerzální konstruktor

Netriviální sebereplikací v celulárních automatech se poprvé zabýval John von Neumann – inspirovaný chováním živých organismů chtěl sestrojit stroj, schopný zkonstruovat svojí vlastní kopii.

Podle von Neumanna by takovýto stroj měl disponovat 3 systémy [9]:

1) Genom, který v sobě uchovává řídící algoritmus,

2) mechanismus, který ze zdrojů dodaných tělem sestrojí svoji kopii

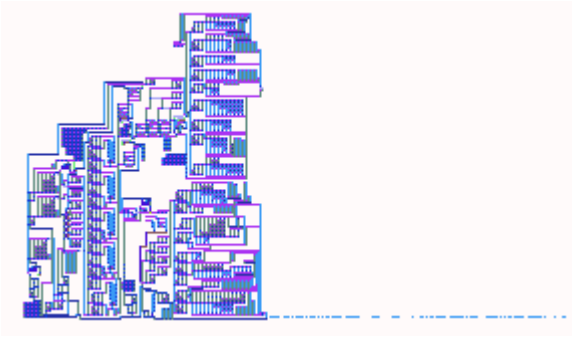
3) a tělo, které má v sobě zakomponované zdroje pro konstrukci a mechanismus pro interpretaci genomu.

Von Neumann vytvořil podle této koncepce v roce 1940 takzvaný univerzální konstruktor, což je složitý model schopný konstruovat libovolný zadaný vzor, tedy i sebe sama. Jeho výpočetní síla je shodná s Turingovým strojem. Skládá se ze tří částí [9]:

1) Paměťové pásky, obsahující popis konstrukce, která má být zkonstruována; v případě sebereplikace obsahuje páska popis automatu samotného,

2) těla konstruktoru, které čte a interpretuje data na paměťové pásce

3) a konstrukčního ramene, řízeného tělem konstruktoru, které se pohybuje po prostoru automatu a zanáší do něj příslušné buňky.



Obr. 4.2 Design von Neumannova univerzálního konstruktoru.

Univerzální konstruktor pracuje s von Neumannovým okolím a používá 29 stavů. Je natolik složitý, že až se současnou technikou ho lze zcela simulovat. Implementace stvořená pod vedením W. Buckleyho v roce 2008 má ve své původní konfiguraci 18 589 buněk v těle a pásku dlouhou 294 844 buněk. Replikace konstruktoru trvá 261 miliard kroků. Právě jeho složitost bráni praktickému využití schopností, které univerzální konstruktor nabízí. Ilustrace jeho modelu je na obrázku 4.2.

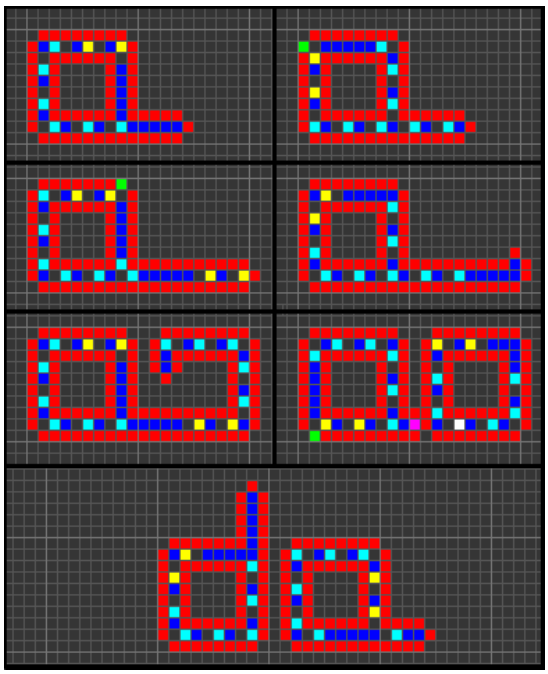
Von Neumannovým konstruktorem začíná výzkum řízené sebereplikace v celulárních automatech. V roce 1968 zjednodušil von Neumannův konstruktor Edgar F. Codd, vystačil si s 8 stavy [10].

4.3 Langtonova Smyčka

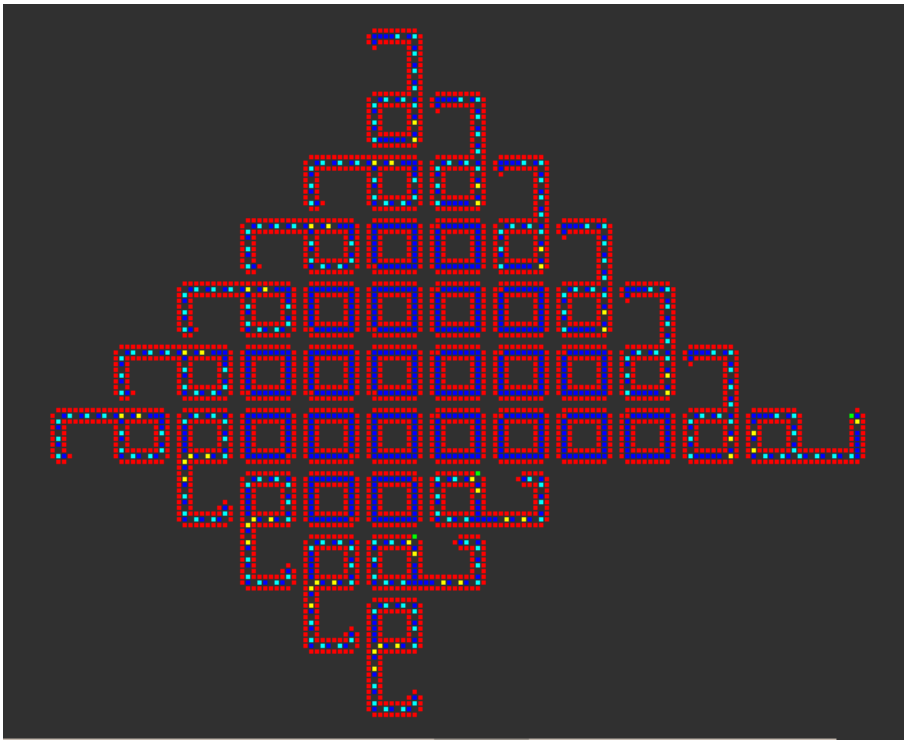
Z Coddova návrhu vyšel Christopher Langton, který podle jedné ze součástí Coddova systému, periodického emitoru, vytvořil Langtonovu smyčku. Langtonova smyčka operuje v automatu používajícím 8 stavů a von Neumannovo okolí, v počáteční konfiguraci má 86 buněk. Je tvořena čtvercovým uspořádáním buněk, v nichž je zakódován genom smyčky, vnitřní a vnější pochvou obalující genom a konstrukčním ramenem. V genomu jsou zakódovány 2 typy instrukcí, mající podobu sekvence 3 buněk: Instrukce pro prodloužení ramene tvoří buňky ve stavech 4 (na obrázku je světle modrý), 0 a 1, buňky instrukce pro stočení instrukce vlevo jsou 7 (na obrázku žlutý), 0 a 1. Buňky genomu cirkulují uvnitř smyčky ve směru proti hodinovým ručičkám; když dorazí na rozhraní těla smyčky a konstrukčního ramene, vzniká jejich kopie, která putuje na konec konstrukčního ramene, kde jsou interpretovány. Rameno se prodlužuje a třikrát se otočí o 90 stupňů vlevo, čímž se vytvoří tělo dceřinné smyčky. Konec ramene potom splyne s jeho kořenem a pomocí signálu reprezentovaným buňkou ve stavu 5 (na obrázku fialový) dojde k zániku spojení mezi mateřskou a dceřinnou buňkou (Langton ho nazval pupeční šňůra).

Obrázek 4.3 ilustruje průběh jedné replikace, která trvá 151 kroků. Mateřská i dceřinná smyčka pokračují v replikaci, přičemž replikace mateřské smyčky je pootočena proti směru hodinových ručiček o 90 stupňů. Smyčka se tak množí dokud nezaplní celý prostor celulárního automatu. Narazí-li konstrukční rameno na tělo jiné smyčky, stáhne se a celá smyčka se stane neaktivní, zmizí informace pro replikaci [11]. Na obrázku 4.4 je znázorněna kolonie Langtonových smyček.

Alternativní řešení pro chování kolonie smyček představil Hiroki Sayama ve své SDSR smyčce (structurally dissolvable self-replicating loop – smyčka schopná rozložit svoji strukturu). Přidáním jednoho stavu k Langtonové smyčce dosáhl chování, kdy se smyčka, která se nemůže už dále replikovat (kvůli vyčerpání celulárního prostoru nebo jeho zabrání jinou smyčkou), sama rozpustí. Celulární prostor tak není zaplněn neaktivními (neboli mrtvými) smyčkami, ale neustále se replikujícími a zanikajícími smyčkami [7].



Obr. 4.3 Průběh replikace Langtonovy smyčky v krocích 0, 8, 29, 34, 109, 127 a 154.



Obr. 4.4 Kolonie Langtonových smyček. Aktivní jsou pouze smyčky v periferních částech kolonie.

Langtonova smyčka patří k nejznámějším sebereplikujícím se smyčkám. Její velký význam spočívá v tom, že po obrovsky komplexním univerzálním konstruktoru ukázala nesrovnatelně jednodušší automat prezentující sebereplikaci. Umožněno to bylo opuštěním konceptu, kterým se řídil von Neumann – sebereplikace je speciálním případem konstrukce. Langtonova smyčka nemá žádné konstrukční nebo výpočetní schopnosti, umí se pouze replikovat.

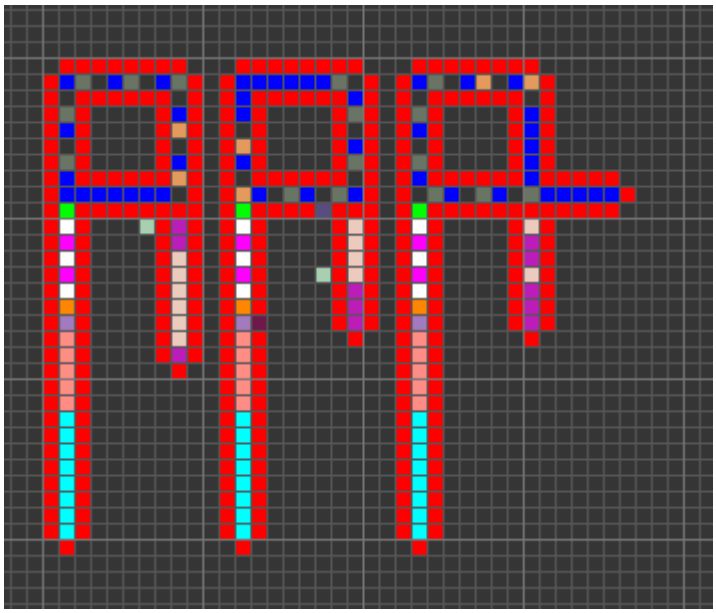
Langtonovu smyčku ještě dále zjednodušil J. Byl, jehož smyčka nemá vnitřní pochvu. Sestává z 12 buněk a používá 5 stavů. Dosud nejednodušší smyčku potom představil Reggia - jeho smyčka nemá ani vnější pochvu. Tvoří ji pouhých 5 buněk a používá 7 stavů [12].

5 Smyčky schopné provádět užitečné úlohy

Kvůli své složitosti není univerzální konstruktor použitelný stroj pro provádění užitečných konstrukcí nebo výpočtů; Langtonova smyčka však podnítila myšlenku obdařit tento jednoduchý sebereplikující se model schopností provádět výpočty či konstruovat.

5.1 Perrierova smyčka

Jean-Yves Perrier použil Langtonovu smyčku jako nosný prvek, ke kterému připojil systém, schopný zpracovávat zadaný algoritmus. Tento systém má stejnou výpočetní sílu jako Turingův stroj [13], sestrojen byl podle takzvaného W-stroje. Ten pracuje se 2 symboly, 0 a 1, prostřednictvím sady 6 instrukcí (`PRINT 0; PRINT 1; MOVE DOWN; MOVE UP; IF 1 THEN; STOP`). V celulárním automatu je Turingův stroj naimplementován pomocí dvou struktur: Seznamu instrukcí a datové pásky. Obrázek 5.1 ukazuje 3 kopie Perrierovy smyčky.



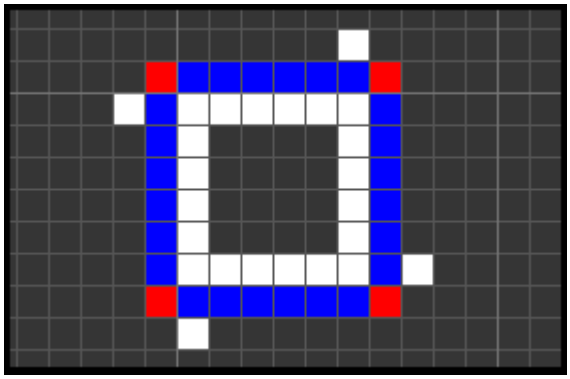
Obr. 5.1 Perrierova smyčka.

Je patrné, že Perrierova smyčka se replikuje pouze v jednom směru celulárního prostoru, do druhého narůstá páska s instrukcemi a daty programu. Replikace probíhá nejprve zkopírováním nosné smyčky. Po dokončení kopírování dojde k přenesení nejprve části reprezentující spustitelný program, poté části reprezentující data mezi mateřskou a dceřinnou buňkou. Potom, co mateřská smyčka dokončí kopírování, rozpojí se spojení s dceřinnou smyčkou a provede se připojený program. Výpočetní schopnosti této smyčky Perrier demonstroval vytvořením programu, kontrolující

uzavřenost závorek. Perrierův automat je schopný nést a provádět libovolný algoritmus za předpokladu dostatku prostoru pro uchování programu a dat, jeho složitost ale zůstala na úrovni, kdy lze chování smyčky plně simulovat. Smyčka s prázdným programem je sestavena ze 158 buněk a vytvoření jedné kopie zabere 235 kroků. Automat používá von Neumannovo okolí a 63 stavů. Jeho omezením je ale kopírování smyčky pouze do jednoho směru, zapřičiněným uložením dat mimo tělo smyčky. Prostorová náročnost také není vyhovující. Jednoduchý program, na kterém Perrier demonstroval možnosti své smyčky, má pásku s instrukcemi delší než 300 buněk – ta je tak více než třicetkrát delší než tělo smyčky. Vykonání celého programu pak zabere více než 35 000 kroků[13].

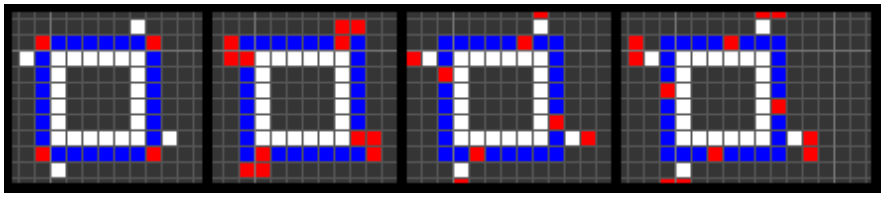
5.2 Tempestiho Smyčka

Gianluca Tempesti vytvořil smyčku s jednoduchým designem, která ale má možnost provádět výpočty či konstrukce. Opět vyšel z podoby Langtonovy smyčky, kterou upravil, aby byla jednodušší a mohla nést data řídící kromě sebereplikace i provádění užitečných úloh. Tempestiho smyčka pracuje s Moorovým okolím, což umožnilo smyčku zjednodušit. Došlo k odstranění vnější pochvy, zůstala jen pochva vnitřní, po které obíhají data řídící replikaci spolu s daty programu. Jednodušší jsou i konstrukční ramena. Ta vyrůstají ze všech čtyřech rohů smyčky zároveň. Jejich růst není stimulován jednotlivými instrukcemi, ale prodlužují se sama. Smyčka řídí pouze jejich otáčení vysíláním signálů, které v pravidelných intervalech dorážejí na konec ramen. Pokud konstrukční rameno narazí na překážku, kterou může být jiná smyčka nebo hranice celulárního prostoru, dojde pouze k rozpuštění tohoto ramene. Ať už dojde ke zkonstruování nové smyčky nebo ne, mateřská smyčka zůstává neustále aktivní a dále v ní cirkulují všechna data původní smyčky. To je výhodné hlavně z hlediska spustitelných programů, které se tak mohou začít provádět po tom, co mateřská smyčka dokončí vytváření svých kopií. Tempesti zmiňuje ještě další výhodu: Pokud dojde ke zrušení smyčky v kolonii, bylo by možné přinutit sousední smyčky, aby na jejím místě vytvořili novou repliku. Existuje zde potenciál opravovat systém tvořený kolonií sebereplikujících se smyček. Kolonie Tempestiho smyček je na obrázku 5.6. Zda dojde k vysunutí konstrukčního ramene řídí buňky, mající funkci brány, indikující, zda se daným směrem může smyčka replikovat. Tempestiho smyčka je tak navržena optimálně pro to, aby zaplnila dostupný celulární prostor svými aktivními instancemi.

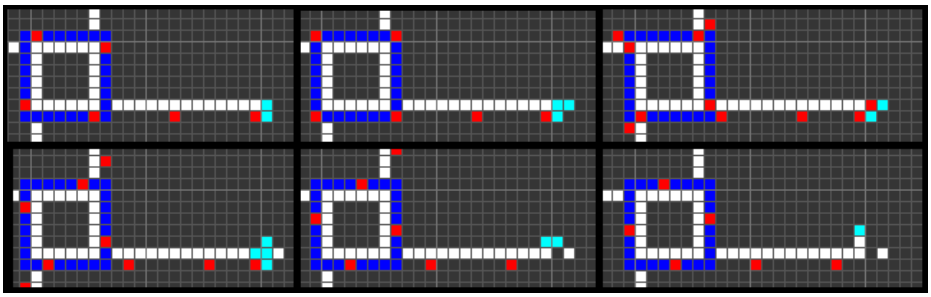


Obr. 5.2 Nejjednodušší podoba Tempestiho smyčky; nenese žádný spustitelný program.

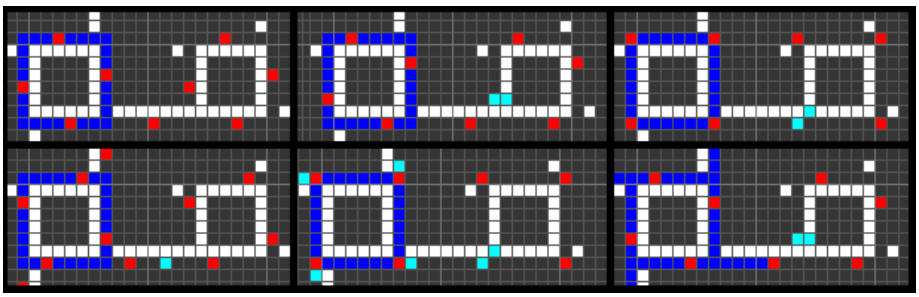
Ve své nejjednodušší podobě, znázorněné na obrázku 5.2, tvoří Tempestiho smyčku čtvercová struktura s rozměry 6x6 buněk, na kterou jsou připojeny další buňky potřebné pro replikaci; celou strukturu tak tvoří 52 buněk. Při její replikaci se využívá šesti stavů: Pět z nich slouží pro replikování struktury smyčky, jeden je datový. Stav 0 (černý) reprezentuje neaktivní pozadí celulárního automatu. Stav 1 (bílý) tvoří vnitřní pochvu a čtyři brány v rozích smyčky. Ty jsou v původní konfiguraci v poloze „otevřeno“, smyčka se tak pokusí do příslušných 4 směrů kopírovat. Ve stavu 2 (červený) jsou buňky signálů, které v počáteční fázi replikace konstruují ramena a potom řídí jejich otáčení. Po prvním otočení ramene řídí jeho růst buňka ve stavu 3 (světle modrý). Stav 3 zkonstruuje vnitřní pochvu dceřinné buňky, po čemž jako signál doputuje k mateřské buňce, indikující tak, že dceřinná buňka je připravená pro kopírování dat, a zahájí jejich kopírování. Stav 4 (zelený) rozpouští pupeční šňůru. Buňky ve stavu 5 (modrý) jsou během replikace pouze pasivně kopírovány. Po rozšíření sady stavů mohou být snadno využity pro uložení dat spustitelného programu. Tempestiho smyčka je také díky jejímu jednoduchému designu jednoduše zvětšitelná, může tak obsáhnout i komplexnější programy[12].



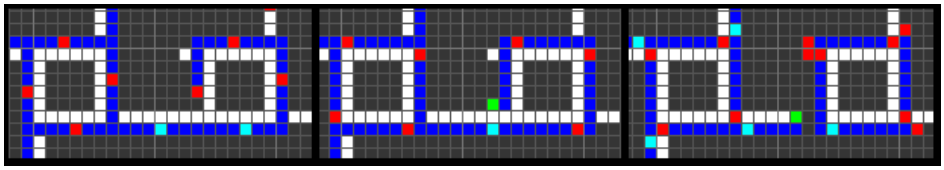
Obr. 5.3 Replikace Tempestiho smyčky, krok 1, 2, 3 a 4. Smyčka se snaží replikovat do 4 směrů. Konstrukční rameno se prodlouží o 1 každé 2 kroky.



Obr. 5.4 Kroky 27 - 32: signál pohybující se každým krokem o 1 buňku doráží na konec konstrukčního ramene a způsobí jeho stočení o 90°. Tyto signály dorážejí na konec ramene v pravidelných intervalech odpovídajících délce smyčky.

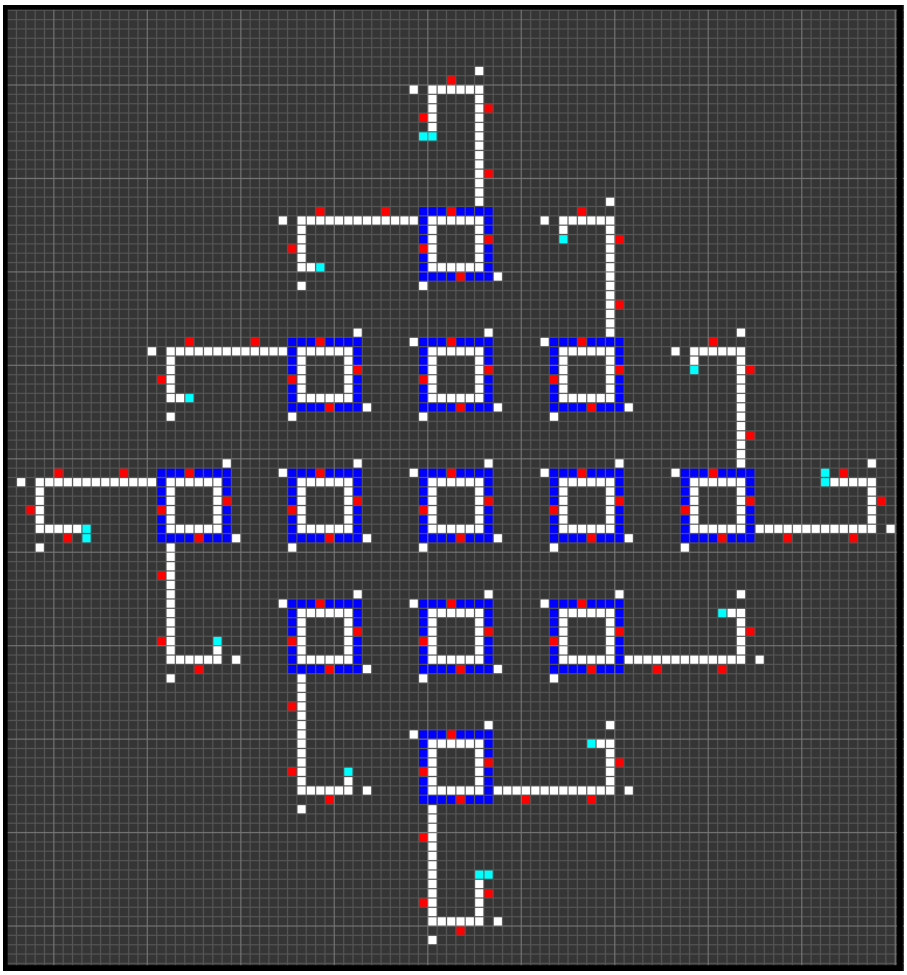


Obr. 5.5 Kroky 67, 68, 70, 72, 77 a 82 - po dokončení konstrukce pochvy funguje stav 3 jako signál, který zahájí kopírování programu z mateřské smyčky na dceřinnou.



Obr. 5.5 Kroky 109, 111 a 113 - Po zkopírování dat programu na dceřinnou smyčku stav 4 rozpustí pupeční šňůru.

Průběh replikace Tempestiho smyčky je zachycen na obrázcích 5.3 – 5.5. Replikace začíná vytrčením konstrukčního ramene, které nejprve vytvoří vnitřní pochvu dceřinné smyčky. Potom, co dojde k uzavření pochvy, vyšle se po pupeční šňůře signál k mateřské smyčce. Ten, poté, co k ní dorazí, odstartuje kopírování dat, obíhající vnitřní pochvu. Když se dokončí kopírování dat, rozpustí se pupeční šňůra a na rozích, ze kterých vyčnívala, se nastaví brány do polohy „uzavřeno“. Dokončení jedné generace kopií trvá 112 kroků.



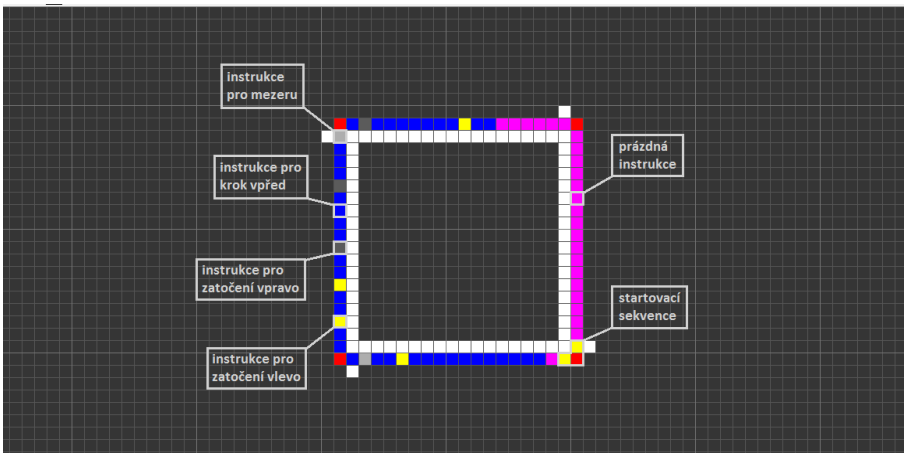
Obr. 5.6 Kolonie Tempestiho smyček. Uvnitř zůstávají buňky aktivní, neustále v nich cirkulují stavy řídící replikaci. Pouze brány v rozích jsou nyní v poloze „zavřeno“.

5.3 Spustitelný program v Tempestiho smyčce

Tempesti vytvořil smyčku, jejíž primární předností oproti Langtonově smyčce je ještě jiná funkcionalita, než pouhá sebereplikace. Možnosti smyčky demonstroval tím, že ji obdařil schopností zkonstruovat písmena „LSL“ (Logic Systems Laboratory) v prázdném prostoru uvnitř smyčky. Pro tento program použil pět datových stavů, přičemž každý z nich se interpretoval jako instrukce při konstrukci písmen. Byly to instrukce pro krok vpřed, otočení doleva, otočení doprava, přerušení struktury mezerou a prázdná instrukce. Data programu nese smyčka o rozměrech 18x18 buněk. K řízení konstrukce použil 330 pravidel. Počáteční konfiguraci jeho smyčky ukazuje obrázek 5.7.

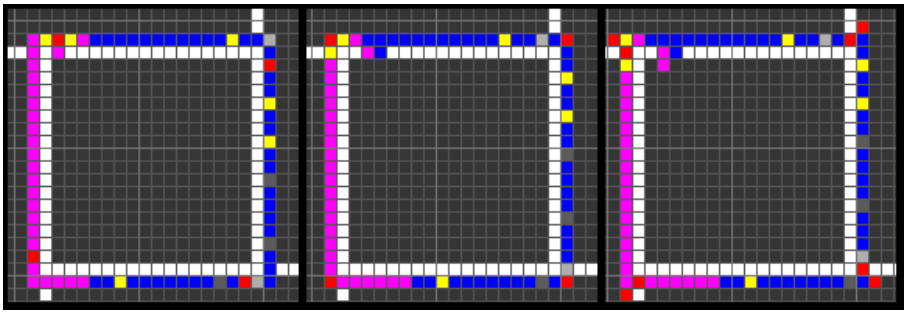
Konstrukce odstartuje v momentě, kdy určitá sekvence buněk dorazí do levého horního rohu smyčky, kde tato sekvence otevře vnitřní pochvu smyčky pro data programu. Když data programu přicházejí k tomuto vstupu, vznikají dvě jejich kopie. Jedna dál obíhá po smyčce, aby mohla být kopírována na dceřinné smyčky, druhá se dostává do prostoru uvnitř smyčky a řídí konstrukci. Kopírování dat do vnitřku smyčky ukončí sekvence prázdných instrukcí, která také uzavře vstup

ve vnitřní pochvě unikátní značkou tvořenou dvěma buňkami, aby už nedocházelo k dalším pokusům o konstrukci. Průběh konstrukce je na obrázcích 5.8 – 5.10.

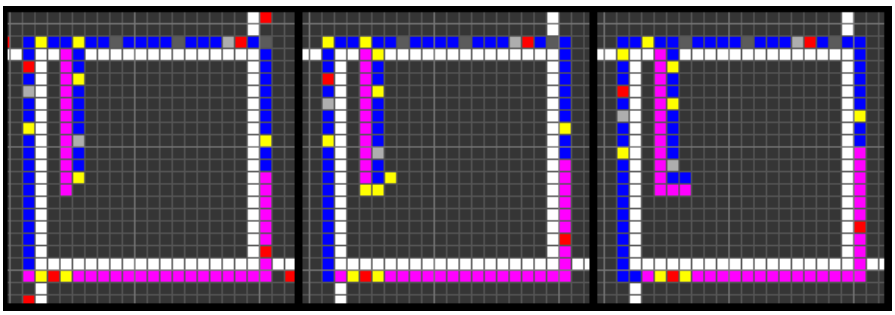


Obr. 5.7 Tempestiho smyčka s programem pro konstrukci nápisu v počáteční konfiguraci.

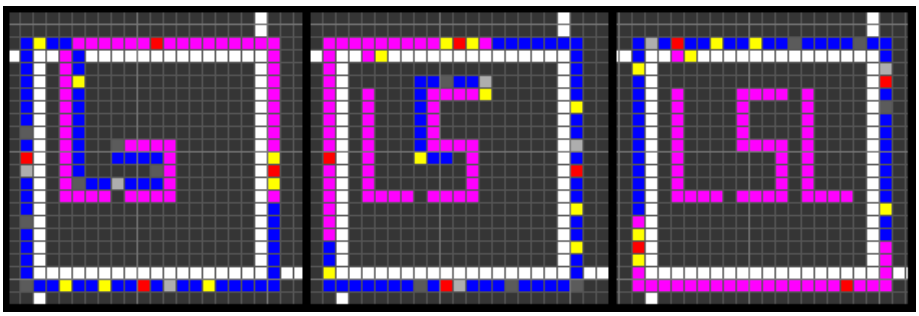
Konstrukce samotná opět využívá konceptu konstrukčního ramene, které se řízením instrukcí prodlužuje a otáčí. Instrukce pro mezeru vytváří přerušení v konstrukčním ramenu, čímž se od sebe oddělí jednotlivá písmena.



Obr. 5.8 Kroky 36, 38 a 39 - inicializační sekvence umožní vstupu dat programu do prostoru smyčky.



Obr. 5.9 Kroky 59, 60 a 61 - instrukce pro otočení dorazila na konec konstrukčního ramene a způsobí jeho otočení.



Obr 5.10 Kroky 85, 104 a 130 - průběh a dokončení konstrukce.

5.4 Zhodnocení smyček provádějící užitečné úlohy

Pro další zkoumání jsem se rozhodl vybrat si pouze jednu z představených smyček. Perrierova smyčka je díky implementaci Turingova stroje teoreticky schopna nést a zpracovat libovolný počítačový algoritmus, to však není nutný předpoklad pro dosažení cíle této práce. Není také schopná provádět jakékoli konstrukce v celulárním prostoru, pouze mění obsah datové pásky. Její nejvýznamnější nevýhoda je však její replikace pouze do jednoho rozměru. Nevhodná je i velikost smyčky nesoucí program – délku pásky instrukcí přesahující 300 buněk považuji za neúnosně velkou.

Tempestiho smyčka disponuje možností nést na sobě data spustitelného programu, tento způsob uchování dat však limituje jeho velikost. Také Tempestim navržené provádění programu v prostoru uvnitř smyčky na něj klade omezení právě velikostí smyčky. Smyčka se však dá flexibilně zvětšovat. Mezi výhody Tempestiho smyčky patří její způsob replikace, kdy se souběžně rozrůstá do čtyřech směrů, a její schopnost vypořádat se s případem, kdy konstrukční rameno narazí na jinou smyčku nebo hranici celulárního prostoru. Dokáže tak efektivně vyplnit aktivními instancemi smyček prostor, který bude konečně velký, ale zároveň dost rozlehlý počtem buněk.

Pro další experimentování jsem si proto zvolil Tempestiho smyčku jako platformu pro spustitelné programy. Cílem je prověřit možnosti smyčky v konstruování a provádění užitečných výpočtů v celulárním prostoru. Při tom bude sledováno zejména narůstání velikosti smyčky se složitostí programu a složitost jejich návrhu.

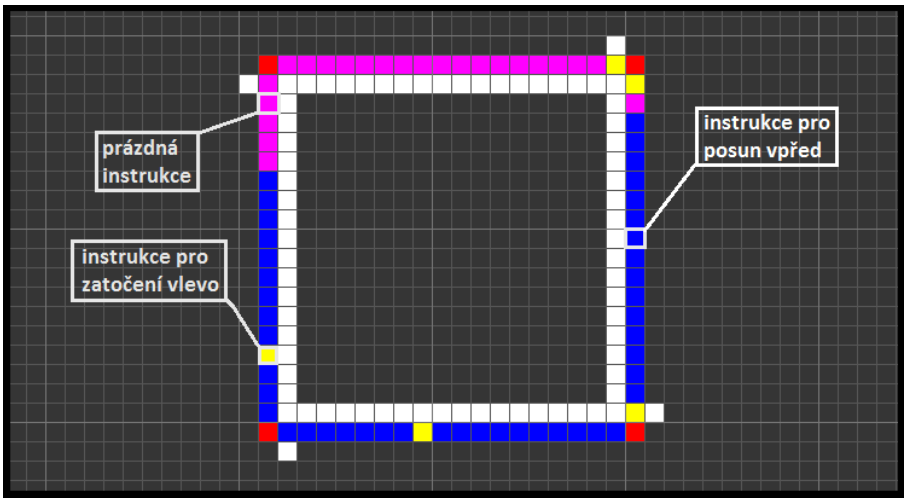
6 Vlastní experimenty

K experimentování jsem používal program Golly[14]. Jde o multiplatformní aplikaci pro simulaci celulárních automatů. Umožňuje uživateli definovat sady pravidel a počáteční konfigurace celulárních automatů. Všechny ukázkové screenshoty v této kapitole i další v této práci byly vytvořeny pomocí této aplikace.

6.1 Vyplnění barevné plochy

Nejjednodušší experiment je konstrukční úloha, kdy dojde k vyplnění čtvercové plochy uvnitř smyčky barvou. Takovéto vybarvování má smysl v prostředí, kde koncept řízení sítě buněk chceme využít hlavně pro obrazové efekty.

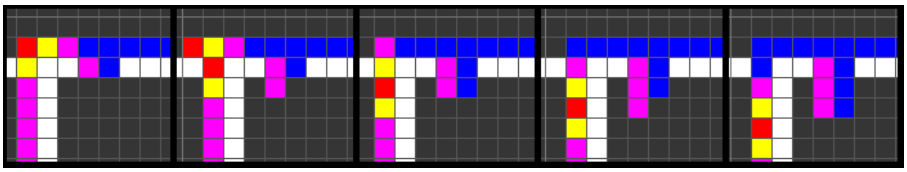
Smyčka s programem pro vybarvení své vnitřní plochy má rozměry 18x18 buněk a používá devět stavů, z toho pět jich je datových. Stav 5 operuje jako prázdná instrukce a slouží k vybarvení plochy. Stav 6 prodlužuje konstrukční rameno, stav 7 ho potom stáčí doleva o 90°. Rozestup buněk ve stavu 7 v počáteční konfiguraci určuje délku hrany čtverce. Stav 8 řídí vyplnění smyčky. Stav 9 rozpouští smyčku kolem vybarvené plochy. Konstrukce začíná stejně jako Tempestiho smyčka konstruující nápis „LSL“. Startovací sekvence otevře v pravém horním rohu smyčky vnitřní pochvu a do prostoru automatu jsou kopírována data programu. Konstrukční rameno stejné jako použil Tempesti sestrojí čtverec, který se vyplní. Po tom, co smyčka vytvoří své kopie a zkopíruje na ně svá data, dojde pro lepší vizuální efekt k jejímu rozpuštění.



Obr. 6.1 Smyčka s programem pro vybarvení své vnitřní plochy v počáteční konfiguraci.

Na tomto místě bych rád popsal způsob práce v prostředí celulárních automatů. Pro řízení chování automatu je třeba definovat množinu pravidel. Při používání Moorova okolí mají pravidla podobu řetězce deseti čísel. První označuje stav dané buňky, dalších osm čísel označuje stavy sousedících buněk. Okolní buňky jsou označovány od severního souseda ve směru hodinových ručiček, neboli posloupností S, SV, V, JV, J, JZ, Z a SZ. Desáté číslo udává, do jakého stavu má daná buňka přejít v následujícím časovém okamžiku. Standardní chování buňky, pro kterou neexistují pravidla, je, že svůj stav nemění.

Na obrázku 6.2 je pravý horní roh smyčky v momentě, kdy startovní sekvence umožnila průnik dat programu do vnitřního programu smyčky a začíná růst konstrukčního ramene směrem dolů. Tabulka 6.1 ukazuje pravidla, která řídí změnu automatu od kroku 19 do kroku 23. Na řádku s číslem kroku je pravidlo, které se použije při přechodu do následujícího kroku.



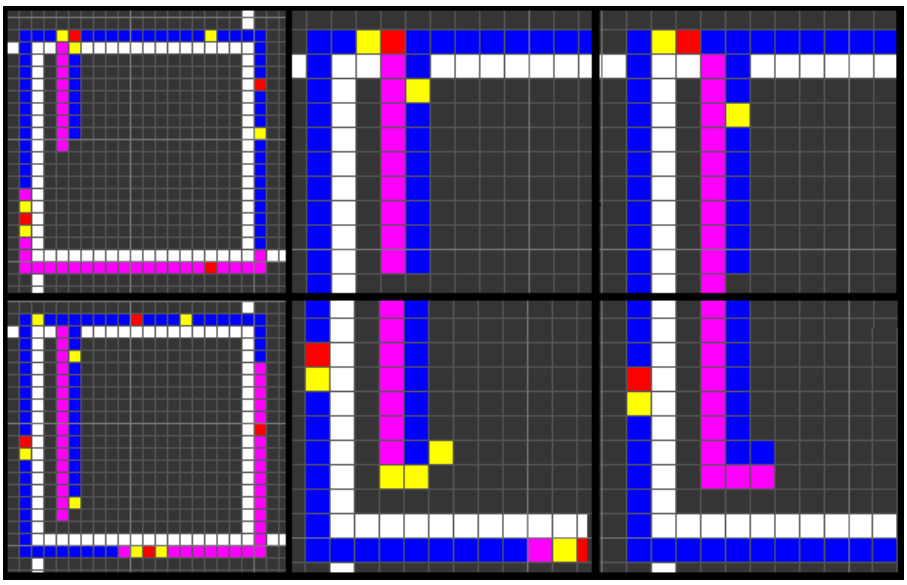
Obr. 6.2 Začátek vyplňování, kroky 19-23.

Konstrukční rameno rameno tvoří stavy 5 a 6, přičemž výsledná konstrukce čtverce stejně jako jeho vyplnění budou tvořeny pouze stavem 5, stav 6 řídí prodlužování konstrukčního ramene. Toto prodlužování je jedna z nejednodušších činností automatu. V každém kroku dochází na konci konstrukčního ramene ke změně jediné buňky. V kroku 19 nejprve vznikne další stav 5 pravidlem [0,5,6,0,0,0,0,0,1,5] následovaný v kroku 20 vznikem stavu 6 vedle něj pravidlem [0,6,1,0,0,0,0,5,5,6]. Po tom, co se v okolí buněk konce ramene již dále nenachází buňky vnitřní pochvy ve stavu 1, řídí identickou konstrukci pravidla [0,5,6,0,0,0,0,0,0,5] a [0,6,0,0,0,0,0,5,5,6].

Krok	Pravidla
19	0,5,6,0,0,0,0,0,1,5
20	0,6,1,0,0,0,0,5,5,6
21	0,5,6,0,0,0,0,0,0,5
22	0,6,0,0,0,0,0,5,5,6
23	-

Tab. 6.1.

Na obrázku 6.3 nahoře je pravý horní roh smyčky v momentě, kdy po konstrukčním rameni začínají putovat signály ve stavu 7, které se na jeho konci interpretují jako instrukce pro otočení ramene doleva (na obrázku 6.3 dole). Obrázek zachycuje i signály 2 a 7 obíhající dál po smyčce. Jimi se v popisu detailněji zabývat nebudu, podrobně je popsal Tempesti [12]. Pravidla použitá v uvedených krocích jsou v tabulce 6.2



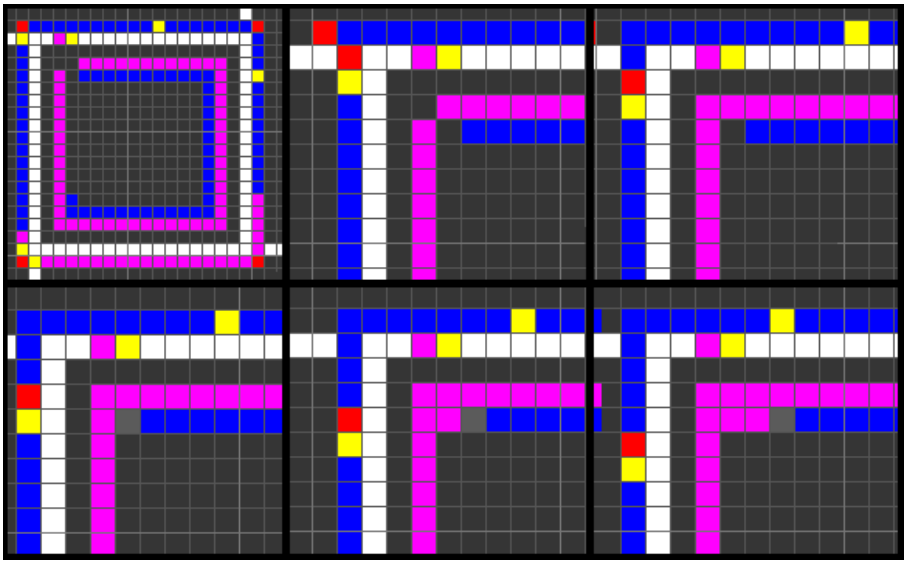
Obr. 6.3 Vybarvování plochy, kroky 34, 35 a 36 (nahoře) a 48, 49 a 50 (dole).

Při posunu signálu po konstrukci dochází vždy v jednom kroku ke dvěma změnám. V kroku 34 se signál 7 posune směrem dolů vznikem nového stavu 7 pravidlem [6,7,1,0,0,6,5,5,5,7] a přepsáním původního stavu 7 na stav 6 pravidlem [7,2,6,1,0,6,5,5,7,6]. V 35. kroku totéž řídí pravidla [6,7,0,0,0,6,5,5,5,7] a [7,6,1,0,0,6,5,5,5,6]. Otáčení, začínající krokem 48, je složitější operace. Nejprve se signál stavu 7 rozvine do tří buněk,které zastaví růst konstrukčního ramene a vytvoří roh. Původní signál 7 přitom zaniká. V dalším kroku se potom buňky v rohu ve stavu 7 přemění do stavu 5, ve kterém už zůstanou, a utvoří se konstrukční rameno s koncem, který roste stejně jako v předchozí fázi.

Krok	Pravidla
34	6,7,1,0,0,6,5,5,5,7 7,2,6,1,0,6,5,5,7,6
35	6,7,0,0,0,6,5,5,5,7 7,6,1,0,0,6,5,5,5,6
36	-
48	5,5,7,0,0,0,0,0,0,7 0,7,0,0,0,0,0,5,5,7 0,0,0,0,0,0,0,7,6,7 7,6,0,0,0,0,5,5,5,6
49	7,5,6,7,0,0,0,0,0,5 7,6,7,0,0,0,0,7,5,5 0,7,0,0,0,0,0,7,6,5 7,0,0,0,0,0,7,6,6,6
50	-

Tab. 6.2.

Po tom, co se konstrukční rameno třikrát otočí doleva, dokončí se konstrukce čtverce a začne vyplňování jeho plochy, které řídí stav 8. Vyplňováni probíhá po řádcích. Na obrázku 6.4 je pravý horní roh smyčky, kde dochází k uzavření čtverce (kroky 114 – 116). Pak se objeví stav 8 (krok 117) a začíná vyplňováni. Příslušná pravidla jsou v tabulce 6.3.

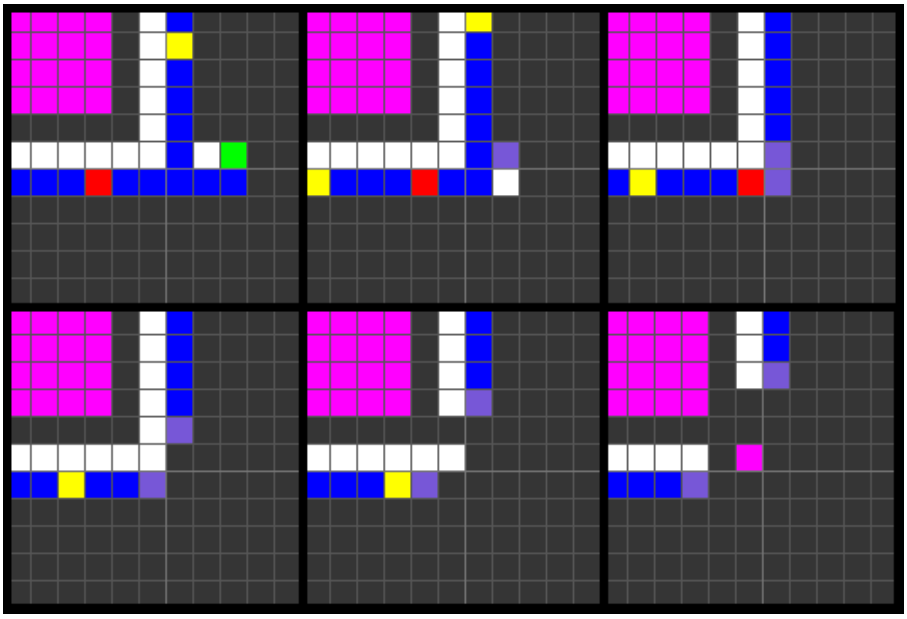


Obr. 6.4 Dokončení konstrukce čtverce a začátek jeho vyplňování - kroky 114-119.

Krok	Pravidla
114	0,0,0,5,6,0,5,0,0,5
115	0,0,0,5,0,5,0,0,0,5
116	0,5,5,6,0,0,5,5,5,8
117	8,5,5,6,0,0,5,5,5,5 6,5,5,6,0,0,0,8,5,8
118	8,5,5,6,0,0,0,5,5,5
119	-

Tab. 6.3.

Po tom, co smyčka zkopíruje data programu na dceřinné smyčky, dojde k rozpuštění pupeční šňůry. Hned na to se začne rozpouštět i samotná mateřská smyčka, jejíž vnitřní plocha je již vybarvena. Rozpouštění řídí dvě buňky ve stavu 9, které z rohů vycházejí opačnými směry. Tato strategie se ukázala jako zbytečně složitá, neboť signál 9 pohybující se po smyčce po směru hodinových ručiček koliduje s ostatními signály obíhající po smyčce. Optimalizace však už nebyla provedena, neboť efektivnost zde není měřítkem. Ukázka rozpuštění pupeční šňůry a začátek rozpouštění smyčky je na obrázku 6.5, příslušná pravidla potom v tabulce 6.4.



Obr. 6.5 Zánik pupeční šňůry řízený stavem 4 – krok 320 a rozpouštění smyčky stavem 9 – kroky 321-325.

Krok	Pravidla
320	1,0,0,4,6,6,6,6,6,9 6,1,4,6,0,0,0,6,6,1 4,0,0,0,0,6,6,1,0,0 6,4,0,0,0,0,0,6,1,0
321	6,6,0,9,1,6,6,1,1,9 6,6,9,1,0,0,0,6,1,9 9,0,0,0,0,1,6,6,6,0 1,9,0,0,0,0,0,6,6,0
322	6,6,0,0,0,9,1,1,1,9 2,1,9,9,0,0,0,6,1,9 9,6,0,0,0,9,2,1,1,0 9,9,0,0,0,0,0,2,1,0
323	6,6,0,0,0,9,1,1,1,9 6,1,1,9,0,0,0,6,1,9 9,6,0,0,0,0,1,1,1,0 1,1,6,9,0,1,1,0,0,0 9,1,0,0,0,0,0,6,1,0
324	1,0,0,0,0,0,9,1,0,5 6,6,0,0,0,9,1,1,1,9 9,6,0,0,0,0,0,1,1,0 1,1,6,9,0,0,0,0,0,0 7,1,1,9,0,0,0,6,1,9 9,1,1,0,0,0,0,7,1,0 1,0,0,1,0,9,7,1,0,0
325	-

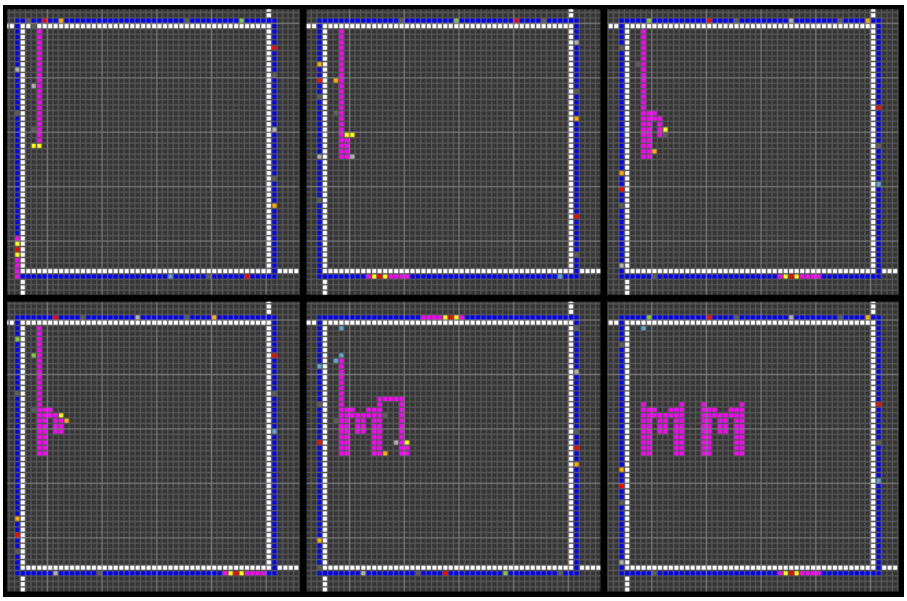
Tab. 6.4.

Tempestiho smyčky mají od sebe po rozkopírování do plochy rozestup o velikosti jedné smyčky. Po rozpuštění smyček je tak rozestup mezi obarvenými čtvercovými plochami příliš velký. Výrazným vylepšením by proto byla úprava mechanismu replikace Tempestiho smyčky tak, aby vzdálenost mezi smyčkami byla kupříkladu poloviční. Dalším možným vylepšením by bylo použití více stavů pro možnost vyplňovat plochy větším počtem barev spolu s propracováním strategie propagace změn barev do dceřinných smyček.

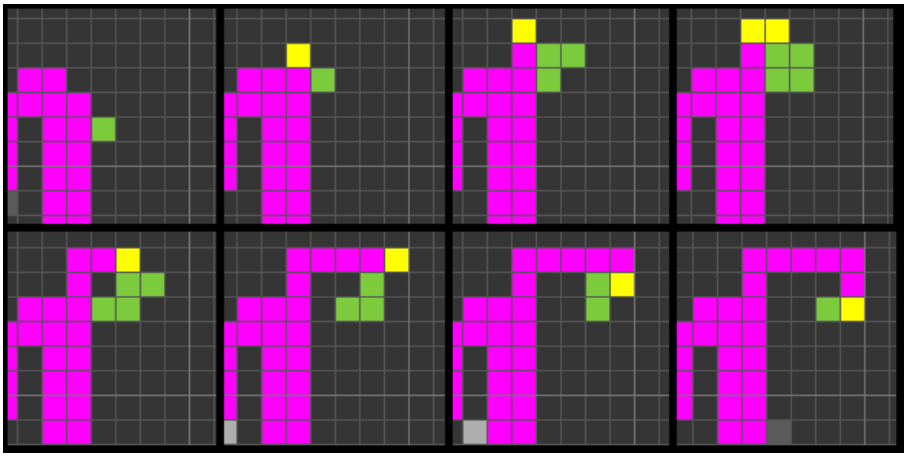
6.2 Konstrukce písmen

Po Tempestiho vzoru jsem se rozhodl zkonstruovat písmena „MM“, abych ozkoušel, jaké konstrukce lze provádět v celulárním prostoru. Na rozdíl od nápisu LSL, kde si konstrukce vystačila s otáčením ramene o 90 stupňů, je tvorba písmen „MM“ složitější.

Na konstrukci je použito 7 datových stavů. Zápis programu se vešel na smyčku o rozměrech 46x46 buněk. Konstrukce začíná opět otevřením vstupu do vnitřní pochvy a kopírováním dat programu do vnitřku smyčky. Konstrukční rameno už má ale jinou podobu, řídí se stejně jako rameno konstruující repliky smyčky. Jeho růst probíhá samovolně, postupuje vpřed o jednu buňku každé dva kroky. Přicházející signály potom mění jeho směr. Konstrukci zobrazuje obrázek 6.3.



Obr. 6.3 Konstrukce písmen, kroky 136, 152, 172, 181, 258 a 357.



Obr. 6.4 Detail konstrukce překlenutí mezi písmeny, kroky 225, 227, 229, 230, 231, 233, 234 a 235. Jde o netriviální konstrukci řízenou jedním stavem. Efektivněji se tak využívají možností CA.

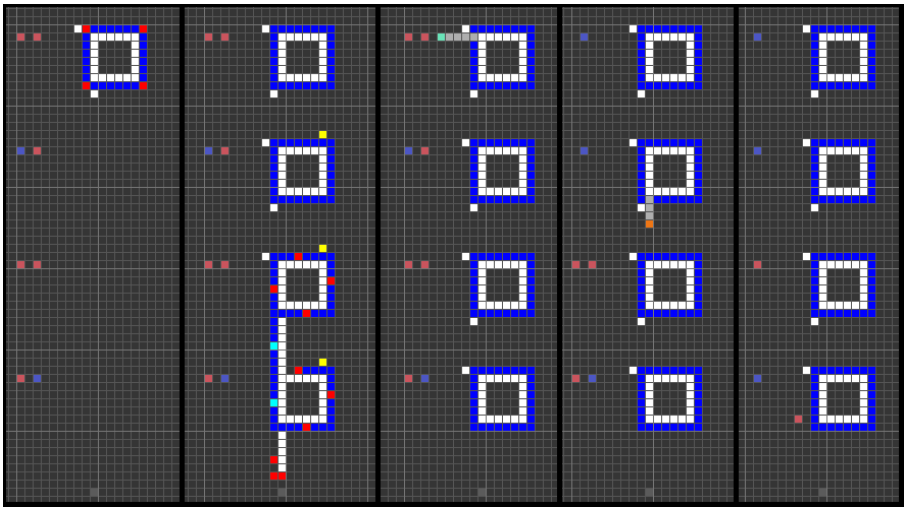
Na tomto automatu se ukázalo, že strategie konstruování řízeného signály, které jsou zapsané přímo ve smyčce a putují do míst konstrukce, není zdaleka optimální. Smyčka nabývá velkých rozměrů, jelikož rozestupy mezi jednotlivými řídícími stavy určují, kdy dorazí na vrchol konstrukčního ramene. Tato strategie je snadno uchopitelná pro návrháře systému, neboť přímo vychází ze systémů běžných v reálném světě. V celulárním prostředí je však nevýhodná pro svou prostorovou náročnost. Netěží z možností, které nabízí celulární automaty. V celulárním automatu pracujícím s Moorovým okolím a 7 datovými stavy lze pro jeho řízení využít 7^9 pravidel, což je přes $40x10^6$ pravidel. Mnou navržený automat jich z tohoto obrovského prostoru využil 697.

Lepší řešení je v tomto případě využití konceptu, kdy se konstrukce vynořuje „sama ze sebe“ - signály by necestovaly do místa konstrukce ze smyčky, ale objevovaly se přímo na místě, kde se odehrává změna. Při této strategii je vhodné, aby počet stavů automatu nebyl příliš limitován. Takový systém ale působí, že zamýšlený design vystupuje z naprostého chaosu. Je proto na návrháři, aby zvolil optimální míru mezi chaotickým chováním a systematickým budováním. Příklad netriviální konstrukce je na obrázku 6.4.

6.3 Binární sčítání

Jedním z cílů je, aby smyčka kromě konstrukcí byla s to provádět i aritmetické operace nebo jiné výpočetní úlohy. Dalším experimentem je návrh binární sčítačky, využívající sebereplikující se smyčky. Design navržený v této práci, zachycený na obrázku 6.5, vyšel z Tempestiho systému, který v celulárním prostorů provádí operace sčítání a násobení. Tempestiho systém je založen na takzvaném kolizním počítání (collision-based computing) [12].

Ve své počáteční konfiguraci se sčítačka skládá s jediné smyčky, v přilehlém celulárním prostoru jsou potom dvě binární čísla. Operace binární sčítačky začíná replikací smyčky jediným směrem. Vytvoří se počet kopií, který odpovídá počtů bitů ve sčítancích. Poslední smyčka při pokusu o replikaci narazí na startovní bit, který potom slouží jako signál propagující se smyčkami až k první smyčce. Ta začne samotné sčítání - pomocí konstrukčního ramene získá nejnižší bity sčítanců, vypočítá jejich součet a do těla smyčky jako signál pošle bit přenosu. Přenos se propaguje do další smyčky, kde se započítá do dalšího bitového součtu. Takto proces sčítání postupuje až k poslední smyčce, u které se zobrazí poslední přenosový bit.



Obr. 6.5 Binární sčítačka, kroky 0, 305, 381, 440 a 505. Ukázka součtu dvou čtyřbitových čísel: 13 (1101) a 7 (0111).

Tento experiment ukazuje, že v celulárním prostředí lze provádět aritmetické výpočty. Operace sčítání není příliš složitá, používá 8 datových stavů a řízena je 263 pravidly. Její prostorová náročnost však není příliš vyhovující. Tempesti tento systém dopracoval dále a realizoval operaci násobení. Ta je ještě více prostorově náročnější, pro násobení dvou N-bitových čísel je zapotřebí 2N smyček. I chování násobičky, která používá přes třicet stavů (univ. konstruktor používá 29), je také výrazně komplikovanější, proto její implementace není součástí této práce.

Tempesti pro implementaci aritmetických operací použil odlišný přístup, než u jeho konstrukční úlohy. Vlastní operace neprobíhá v prostoru uvnitř smyčky, ale mimo něj. První nevýhoda tohoto přístupu je rozpor s mým záměrem – distribuovat do celulárního prostoru jednotky provádějící výpočty. Další nevýhodou je, že smyčka při provádění operace musí přijímat a vysílat signály o přenosech, což vytváří problémy se synchronizací činnosti smyček. V mé implementaci mě tyto problémy dovedly ke změně chování smyčky – odstranil jsem signály řídící replikaci smyčky, které jinak obíhaly ve smyčce. Sčítačka byla navržena tak, aby se výpočet provedl právě jednou, nepřipravujeme se proto o možnost opravy nebo znovuprovedení replikace smyčky.

7 Závěr

V této práci jsem zhodnotil sebereplikující se smyčky jako platformu pro provádění užitečných úloh. Ze dvou kandidátů pro provádění testování možností smyček – Temestiho a Perrierovy smyčky jsem vybral Tempestiho smyčku. Její výhody jsou efektivní mechanismus kopírování v celulárním prostoru, schopný vypořádat se s překážkami v podobě hranic celulárního prostoru nebo kolize dvou smyček, a možnost provádět kromě výpočetních i konstrukční úlohy.

V Tempestiho smyčce jsem implementoval tři úlohy. První z nich bylo vyplnění plochy uvnitř smyčky. Výchozím bodem při návrhu tohoto automatu pro mě byla Tempestiho smyčka obsahující program pro konstrukci písmen „LSL“. Po tom, co mnou navržená smyčka provedla program, který uvnitř smyčky zkonstruoval čtverec a vyplnil ho, dochází k jejímu rozpuštění. Pro zkonstruování a vyplnění čtverce bylo třeba k původnímu setu pravidel Tempestiho smyčky přidat dalších 64 pravidel. Vezmeme-li v úvahu, že chování automatu se změnilo poměrně značně, cena této úpravy byla velmi malá. Pro rozpuštění smyčky bylo zapotřebí 357 pravidel, přičemž jsem při jejich zápisu používal proměnné. Počet pravidel, se kterými automat skutečně pracuje je tedy vyšší. Proces rozpouštění by se ale výrazně zjednodušil zvolením jednodušší strategie.

Další úlohou byla konstrukce písmen „MM“ uvnitř smyčky. Využil jsem stejné strategie jako Tempesti, kdy konstrukci postupně vytváří konstrukční rameno, jehož konec je řízen přicházejícími signály. Ty jsou od začátku zapsány v programové části smyčky. Tato strategie vede k nárůstu velikosti smyčky a zdaleka tak není vhodná pro složitější konstrukce, je vhodné hledat strategie jiné. V šesté kapitole byl naznačen jeden z možných postupů. Výchozí modelem pro úpravy byla Tempestiho smyčka bez programu o rozměrech 46x46 buněk, pro řízení konstrukce bylo její sadě pravidel přidáno 697 pravidel dalších. Implementaci této úlohy nepovažuji za náročnou.

Třetí úlohou byla konstrukce binární sčítačky podle Tempestiho návrhu. V tomto automatu se úloha neprovádí uvnitř prostoru sebereplikujících smyček, což považuji za značnou nevýhodu, jelikož zde zaniká možnost paralelního provádění výpočtů v celém prostoru celulárních automatů. Bylo by vhodné najít cestu, jak výpočty umístit dovnitř smyčky. S tím jsou však spojeny nová úskalí, zejména s distribuci operandů do prostoru smyček a odečítání výsledků. Automat vznikl z prosté Tempestiho smyčky bez programu, přidal jsem 261 pravidel.

V této práci jsem naimplementoval novou funkcionalitu do Tempestiho smyček. Doufám ale, že jejím přínosem bude i podrobné vysvětlení pravidel celulárního automatu a jejich souvislosti s chováním automatu. Detaily těchto principů totiž dosud nebyly publikovány. K implementaci sebereplikujících se smyček buch rad zmínil několik obecných poznatků v tomto závěru. Tvorba sady pravidel vyžaduje přípravu. Je vhodné si detailně naplánovat průběh konstrukcí a zamýšlené chování automatu. Díky programu *Golly* si lze ulehčit zejména návrh dílčích fází konstrukce. Možnosti v konstruování jsou velké, návrh i složitějších strojů není nikterak neúnosně náročný na implementaci. Provádění aritmetických operací je možné, při složitějších operacích ale značně narůstá komplexnost i prostorová náročnost automatu. Stejně jako u klasického programování lze pracovat systematicky (využívat signály cestující na místo provádění změny) a znovupoužívat příslušný kód, to vše na úkor prostorové náročnosti, nebo navrhovat unikátní konstrukce. V obou případech často přinese zjednodušení zvýšení počtu stavů. Budoucí systémy, jež by měly jako předobraz celulární automat, by měly disponovat možností počet stavů navyšovat alespoň do řádů desítek.

Tato práce poskytuje základní přehled o možnostech sebereplikujících se smyček v celulárních automatech. Případný další výzkum by určitě přinesl mnoho nových zajímavých poznatků. Zejména v oblasti provádění výpočtů je zatím nejisté, kam možnosti smyček sahají. Pokud by si smyčka neměla nést operandy, musí být řešeno jejich přijímání. Stejně tak je otázkou, jak v celulárním prostoru naložit s výsledky. Dalším možným předmětem bádání by mohla být možnost ovlivnit rozestupy smyček či jejich tvar. Díky blízké souvislosti principů celulárních automatů se systémy reálného světa je inspirace pro další výzkum na každém kroku a věřím, že experimentování v celulárních automatech odhalí nové pravdy.

Literatura

[1] ILACHINSKI, A.: *Cellular automata: A discrete universe.* 2nd edition. Singapore: World Scientific Publishing Co. Pte. Ltd., 2002. ISBN 981-02-4623-4.
[2] SEKANINA, L.: *Evolvable components: From theory to hardware implementations*. Springer, 2004. ISBN 978-3-540-40377-7.
[3] WOLFRAM, S.: *A new kind of science.* Wolfram Media, Inc, 2002. 1280s. ISBN 1-57955-008-8.
[4] LAY, J.: *A Land Use Change Study Using Cellular Automata*. [online]. C2000. <http://www.gisdevelopment.net/aars/acrs/2000/ts12/ts12003.asp>.
[5] SIPPER, M.: *Evolution of Parallel Cellular Machines - The Cellular Programming Approach.* Lecture notes in Computer Science, vol. 1194, Springer, 1997.
[6] *Conway's Game of Life* [online]. Last revision 5th of May 2010 [cit. 2010-05-08]. <http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life>.
[7] SAYAMA, H.: *Spontaneous evolution of self-reproducing loops on cellular automata*, Unifying Themes in Complex Systems: Proceedings of the Second International Conference on Complex Systems, vol. 2, p. 363-374, Westview Press, 2003.
[8] LOHN J. D.: *Cellular space model of self replicating systems. Lectures on mathematics in the life sciences, Volume 26.* [online]. [cit. 2010-05-08]. <http://books.google.cz/books?id=QfMS-SCCJskC>.
[9] VON NEUMANN, J. - BURKS, A. W.: *Theory of Self-Reproducing Automata*. 1966.
[10] CODD, E. F.:*Cellular Automata.* New York: Academic Press, 1968.
[11] LANGTON, C. G.: *Self-reproduction in cellular automata.* Physica D: Nonlinear phenomena 10. 1984.
[12] TEMPESTI, G.: *A self-repairing multiplexer-based FPGA inspired by biological processes.* Lausanne, 1998. Rigorózní práce na Computer science department na Swiss federal institute of technology. Vedoucí práce Prof. Daniel Mange.
[13] PERRIER, J.: *Toward a viable, self-reproducing universal computer.* Advances in Artificial Life. Lecture notes in Computer Science, vol. 1674, Springer, 1999.
[14] TREVORROW A. - ROKICKI T. *Golly.* [počítačový soubor]. Verze 2.1. 2009.

Seznam příloh

Příloha 1: CD

Příloha 1

Obsah CD:

`\Golly`

V této složce se nachází program Golly, spustitelný souborem *Golly.exe.* Slouží k simulaci celulárních automatů. Návod k použití je v souboru *Navod.pdf.*

`\Golly\Rules`

V této složce jsou soubory s pravidly pro celulární automaty. Mnou vytvořená pravidla jsou v souborech *addition.table* – pro smyčku provádějící binární sčítání, *mm.table* – pro smyčku konstruující písmena „MM“ a *square.table* – smyčka vyplňující svoji vnitřní plochu. Svojí vlastní práci jsem v těchto souborech vyznačil poznámkami **#muj kod:**, příp. **#konec meho kodu**. Dále se v této složce nachází soubory *addition.colors*, *square.colors* a *mm.colors*, v nichž jsou konfigurovány barvy stavů příslušných automatů v programu Golly.

`\Golly\Patterns\Loops`

Zde se nachází soubory s popisem počátečních konfigurací celulárních automatů. Pro mnou navržené automaty to jsou soubory *bc-addition.rle*, *bc-square.rle* a *bc-mm.rle*.