VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

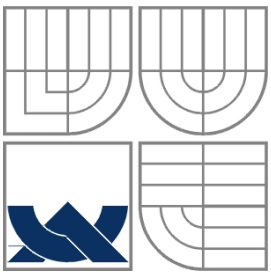
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

SEBEREPLIKACE V CELULÁRNÍCH SYSTÉMECH

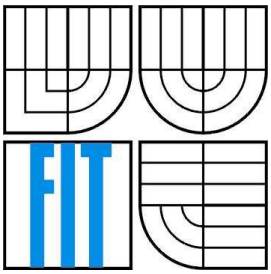
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Tomáš Komenda
AUTHOR

BRNO 2009



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 SYSTÉMECH

SELF REPLICATION ON CELLULAR SYSTEMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Tomáš Komenda

VEDOUCÍ PRÁCE
SUPERVISOR

doc. Ing. Lukáš Sekanina Ph.D.

BRNO 2009

Zadání diplomové práce/4915/2008/xkomen03

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2008/2009

Zadání diplomové práce

Řešitel: **Komenda Tomáš, Bc.**
Obor: Počítačové systémy a sítě
Téma: **Sebereplikace v celulárních systémech**
Kategorie: Umělá inteligence

Pokyny:

1. Seznamte se s problematikou celulárních automatů a jejich aplikacemi.
2. Vypracujte studii na téma celulární automaty, celulární výpočetní platformy a sebereplikace.
3. Navrhněte a implementujte simulátor 2D celulárního automatu, který bude vhodný pro demostraci sebereplikace a bude umožňovat použití globální informace pro řízení funkce buněk.
4. Pomocí vytvořeného simulátoru demonstrujte sebereplikaci zvolených objektů (např. Langtonovu smyčku apod.) a využití globální informace.
5. Shrňte dosažené výsledky.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části diplomového projektu je požadováno:

- Splnění prvních 2 bodů zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Sekanina Lukáš, doc. Ing., Ph.D.**, UPSY FIT VUT
Datum zadání: 22. září 2008
Datum odevzdání: 26. května 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

Abstrakt

Tato práce se zabývá celulárními systémy a jejich využitím pro sebereplikaci datových struktur. Samotné odvětví celulárních automatů je velmi zajímavou a inspirativní oblastí, která se v současnosti ukazuje jako velmi vhodné prostředí pro simulaci různých jevů. Jedním z těchto jevů může být například i umělý život nebo sebereplikující se struktury, které přenáší jistou užitečnou informaci nebo provádí potřebné výpočty. V práci jsou podrobně popsány celulární automaty a jejich dělení. V oblasti sebereplikace se zaměřuji na Langtonovy smyčky, Coddův Automat, Bylovy smyčky, Chou-Reggia smyčky, Tempesti smyčky, Perrierovy smyčky, SDSR smyčky, Evoloop a Sexyloop. Součástí práce je návrh způsobu replikace Bylových smyček pomocí změny pravidel celulárního automatu a doplnění jejich funkčnosti o schopnost uvolnit svůj prostor po splnění úkolu ostatním smyčkám. V práci jsem se zabýval i evolučním návrhem pravidel celulárního automatu.

Klíčová slova

Celulární automat, sebereplikace, sebereprodukce, Von Neumannův automat, Cell Matrix, Langtonovy Q-smyčky, SR-Loops, SDSR-Loops, Sayama, Sexyloop, Evoloop, Bylova smyčka, Coddův automat, Chou-Reggia smyčka, Tempesti smyčka, Perrierova smyčka, urychlená Bylova smyčka, rozpustná urychlená Bylova smyčka.

Abstract

This thesis deals with cellular systems and their applications to self - replication data structures. The sector cellular automats is a very interesting and inspiring area which seems now as a very suitable environment for the simulation of various phenomen. One of these phenomen may be, for example, artificial life or self- replication the structure, which transmits some useful information or carry out the necessary calculations. In this thesis is detailed subscribe cellular automats and their division. It focuses on Langton’s loop, Codd‘s automata, Byl‘s loop, Chou-Reggia loop, Tempesti loop, Perrier loop, SDSR loop, Evoloop and Sexyloop. Part of the work is to accelerate replication Byl‘s loops through change to the rules of cellular automat and the addition of functionality to the ability to release space to the completion of replication loops. At thesis, I also dealt with the evolutionary design of the rules of cellular automata.

Keywords

Cellular automata, self replication, self reproduction, Von Neumann's automata, Cell Matrix, Langton, Self-Reproducing loops, SR-Loops, SDSR-Loops, Sayama, Sexyloop, Evoloop, Byl's loop, Codd's automata, Chou-Reggia loop, Tempesti loop, Perrier loop, fast Byl's loop, fast SD-Byl's loop.

Citace

Tomáš Komenda: Sebereplikace v celulárních systémech. Brno, 2009, diplomová práce, FIT VUT v Brně.

Sebereplikace v celulárních systémech

Prohlášení

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

……
Tomáš Komenda
25.5.2009

Poděkování

Zde bych chtěl vyjádřit poděkování svému vedoucímu doc. Ing. Lukáši Sekaninovi Ph.D. za konzultace a rady, které vždy ochotně poskytnul.

Obsah

1 Úvod

Sebereplikace v celulárních systémech je velmi zajímavou tématikou, kterou se již zabývalo mnoho zkušených vědců [1],[2]. Pod pojmem sebereplikace si můžeme představit například robota, který vytváří své vlastní kopie [9]. Každá z nich je schopna dále stavět své další a další potomky. S každou další generací se masivně zvyšuje míra uplatněného paralelismu. Jeden takový jedinec by mohl vytvářet celé kolonie robotů a na těžko dostupných místech by mohl plnit mnoho dříve nemožných úkolů. Nespojujme však sebereplikaci pouze s roboty, právě naopak. Tato práce se zaměřuje na sebereplikující se datové struktury v celulárních systémech. Pokud je taková struktura navržena správně a dosahuje jisté meze složitosti, je schopná provádět užitečné výpočty nebo přenášet netriviální informaci. V neposlední řadě se tato problematika přímo dotýká umělého života a jeho simulací. Již jednobuněčné organizmy se množí, vznikají potomci stejného typu, kteří od rodičů získávají genetickou informaci. Pokud k těmto poznatkům přidáme i názory některých vědců, kteří říkají, že podstatou života není materiální forma, která jej reprezentuje, ale právě informace, kterou v sobě živý jedinec uchovává, přenáší a předává dále, můžeme říct, že za určitých podmínek by bylo možno v těchto systémech úspěšně realizovat umělý život [1].

Základní myšlenkou této práce je urychlení sebereplikace. Víme, že určité struktury jsou svou sebereplikací užitečné. Zrychlení replikace by tedy mohlo přinést mnoho pozitivních faktů. Urychlení replikace, které je v této práci navrženo, je založeno na změně pravidel celulárního automatu během jeho činnosti. Podnět k urychlení tohoto typu přinesl vývoj takzvaných polymorfních hradel [20]. Jedná se o komponenty, které mění logickou funkci v závislosti na externím stimulu, například ve formě změny napájecího napětí, teploty, stavu dopadajícího světla a podobně. Díky těmto schopnostem by bylo možné vytvořit celulární systém, který by na základě změny externího prostředí pozměnil pravidla, jimiž je řízen.

Tato práce se ve svých prvních kapitolách zabývá celulárními systémy a jejich problematikou. Již dnes nalézají celulární automaty uplatnění v mnoha odvětvích. Jejich působnost sahá například do generování pseudonáhodných čísel, generování testovacích vektorů, simulací chování plynu, simulací feromagnetismu, simulací růstu krystalů, modelování ekonomických procesů, modelování diferenciálních rovnic, simulací fyzikálních jevů i umělého života a sebereplikace [6]. Tyto zdánlivě jednoduché systémy jsou schopny dosahovat díky svému emergentnímu chování velmi pozoruhodných výsledků.

V následující části je popsána architektura Cell Matrix, která je založená na celulárních systémech a jeví se velmi slibnou pro využití v nanotechnologii.

Dále se věnuji sebereplikujícím se strukturám. Zaměřuji se na celou řadu těchto elementů počínaje jednoduchými binárními strukturami až po velmi složité smyčky, které jsou schopny evolučního chování a vzájemného křížení. Celá práce se opírá o Langtonovy a Bylovy smyčky.

V rámci této práce jsem navrhl urychlení replikace Bylovy smyčky na principu změny pravidel v průběhu replikace. Popis mechanismu urychlení je obsahem další kapitoly této diplomové práce. Tuto smyčku jsem také obohatil o schopnost rozpustit svoji strukturu po dosažení stavu, kdy již není pro vývoj komunity těchto struktur užitečná. Uvolní tak potřebné místo novým smyčkám. Tato modifikace je zde také podrobně popsána a jsou zde uvedeny výsledky experimentů.

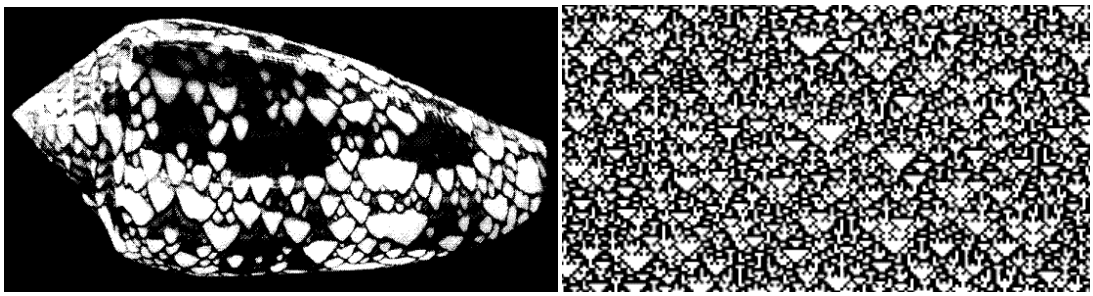
V závěrečných částech se zaměřuji na popis simulátoru celulárního automatu, který jsem implementoval v rámci této práce, a který umožňuje demonstraci dosažených výsledků. Také zde popisuji pokus o evoluční návrh pravidel některých struktur popsaných v této studii.

V rámci semestrálního projektu jsem vytvořil studii o celulárních automatech a architekturách založených na těchto systémech. Také jsem se zaměřil na Langtonovu smyčku a podrobně jsem prostudoval její chování a výsledky popsal do studie. V rámci diplomové práce jsem pokračoval ve studii dalších struktur provádějících svou replikaci, implementoval jsem simulátor celulárního automatu a navrhl pomocí něj urychlení replikace Bylovy smyčky a její rozpustnou formu. Dosažené výsledky jsem popsal v této práci. Nad rámec zadání jsem se pomocí evoluce pokusil navrhnout pravidla některých struktur popsaných ve studii. Výsledky jsem rovněž uvedl v diplomové práci.

2 Celulární automaty

Celulární automat je paralelní výpočetní model s lokální interakcí výpočetních elementů. Bylo pozorováno, že vykazuje emergentní chování. To znamená, že zdánlivě složité chování, které je patrné již na první pohled, způsobují „jednoduchá“ lokální pravidla (v případě celulárních automatů lokální pravidla jednotlivých buněk). Celulární automaty se dnes používají v mnoha odvětvích. Vytváří například vhodné simulační prostředí pro různé jevy. Nejznámějším počinem v tomto směru je bezesporu hra Life. Využívají se však i k simulaci šíření lesních požárů, nakažlivých nemocí, autodopravy, polární záře, magnetismu, elektrických obvodů a v mnohých dalších oborech [5],[6]. Celulární automat je možné využít i k výpočtům a řízení. Lze dokázat, že jisté varianty celulárních automatů mají stejnou výpočetní sílu jako Turingův stroj [1].

Celulární automaty jsou velmi spjaty s biologií a vykazují zajímavé vlastnosti. Vezměme v úvahu druhý zákon termodynamiky. Ten říká, že v uzavřeném systému entropie roste. Entropie je mírou neuspořádanosti a její růst je v rozporu s procesem biologické evoluce, která naopak vytváří vysoce organizované struktury [1]. Tento protiklad je možné pozorovat právě u některých typů celulárních automatů, které pracují na hranici mezi chaosem a řádem. Například von Neumann tvrdil, že existuje jistá kritická mez složitosti, pod kterou jsou procesy syntézy degenerativní, zatímco nad ní má syntéza při správné organizaci explozivní charakter. Langton pak zastával názor, že na těchto mezích může existovat život. Díky těmto vlastnostem mohou být celulární automaty používány pro simulaci umělého života [1]. Podobnosti se skutečným životem si můžeme všimnout na obrázku 1, kde v levé části je zobrazena skořápka mořského tvora, jež je pigmentována vzorem, který je podobný výstupním strukturám některých celulárních automatů. V pravé části obrázku je znázorněna struktura produkována celulárním automatem. Obrázek 1 je převzat z [5].



Obrázek 1.: Skutečný život (skořápka mořského tvora) vs. výstup celulárního automatu

2.1 Obecný popis celulárního automatu

Celulárním automatem (buněčným automatem) je myšlen paralelní a dynamický systém výpočetních uzlů (buněk), které pracují v diskrétním čase.

Teoreticky se jedná se o matematický model, který popisuje strukturu buněk v *N*-rozměrném prostoru. Každá buňka nabývá jednoho z konečného počtu stavů. Tento stav je dán takzvanou přechodovou lokální funkcí. Tuto funkci si můžeme představit jako množinu pravidel, pomocí které je vypočten následující stav buňky. Tedy argumentem lokálních funkcí bývá stav vyšetřované buňky a stavy sousedních buněk. Buňka tak má informaci nejen o svém vlastním stavu, ale i o stavu svého blízkého okolí. Výstupem lokální funkce je nový stav vyšetřované buňky [1]. Pokud je tato funkce pro všechny buňky automatu stejná, označujeme jej jako *uniformní*. Existují však i celulární automaty, které pro různé buňky disponují různými funkcemi, takové pak označujeme jako *neuniformní* [6].

Počet sousedních buněk, tedy velikost sousedství, pak závisí na definici konkrétního automatu. Teoreticky můžeme uvažovat nekonečně velký celulární automat, tedy složený z nekonečného množství buněk. V reálných implementacích jsme však omezeni technickými prostředky a musíme tedy uvažovat i chování systému na okrajích. Tato situace se řeší několika způsoby. Jedním z nich může být úprava lokálních funkcí okrajových buněk. Dalším často používaným způsobem je, že přiřadíme již neexistujícím buňkám (buňkám za okrajem) určitý neměnný stav nebo propojíme okraje automatu. Propojením vytvoříme v jednorozměrném automatu smyčku, u dvourozměrného automatu anuloid [1].

Obecná definice celulárního automatu může být velice široká a záleží na tvůrci samotném, jaký automat vlastně implementuje. Pro obecný (učebnicový) typ celulárního automatu jsou však charakteristické vlastnosti paralelismu, homogenity a lokality [1].

Paralelismem máme na mysli skutečnost, že výpočet hodnot stavů všech prvků celulárního automatu probíhá současně. Tato vlastnost je velmi výhodná pro využití paralelních systému a tím i dosažení velké rychlosti při běhu simulace. Naopak při běhu na běžných strojích se musí paralelismus simulovat.

Homogenita v původním smyslu vyjadřovala totožnost přechodové lokální funkce pro všechny buňky automatu. Tato skutečnost ale nemusí být dodržena u neuniformních celulárních automatů, proto je tato vlastnost spíše formální vlastností při popisu jakéhosi vzorového celulárního automatu.

Lokalita pak říká, že nový stav závisí pouze na stavu vyšetřované buňky a na stavu jejího definovaného sousedství. Tato vlastnost je stejně jako v případě homogenity spíše formální vlastností učebnicového modelu. Připomeňme, že smyslem této práce je pokusit se rozšířit celulární systém o globální informaci umožňující změnit celkový vývoj automatu.

2.2 Formální definice celulárního automatu

Formální definice celulárního automatu, která vychází z [2], se týká automatu, který vykazuje vlastnosti paralelismu, homogenity a lokality. Hodnoty stavů i času jsou hodnoty z diskrétní množiny.

Uvažujme tedy automat, jehož součástí je množina buněk, které nabývají určitých stavů. Tyto stavy můžeme chápat jako podmnožinu diskrétních hodnot Z^D v *D*-dimenzionálním prostoru. Každá buňka má definované své sousedství. Šablonu sousedství, která je použitelná pro každou buňku automatu, pak definujme jako $N = \{z_0, z_1, \dots, z_{n-1}\}$, kde pro z_i platí, že $z_i \in Z^D$, kde $i \in \{0, 1, \dots, n-1\}$. Stav konkrétní buňky v daném čase *t* pak označme jako $s_t(z)$ a platí pro něj, že patří do konečné množiny stavů buňky Q. Tedy můžeme napsat, že $s_t(z) \in \mathrm{Q}$. Dále si definujme lokální přechodovou funkci jako $\Delta: \mathrm{Q}^n \rightarrow \mathrm{Q}$. Nyní můžeme vývoj tohoto automatu v jednotlivých krocích času zapsat následovně.

$$s_{(t+1)}(z) = \Delta(s_t(z_0 + z), s_t(z_1 + z), \dots, s_t(z_{n-1} + z)) \qquad (1)$$

Z formální definice je tedy patrné, že nový stav dané buňky skutečně závisí pouze na stavu vyšetřované buňky a na stavu okolních buněk. Jak již bylo uvedeno v předcházejících částech, existují i neuniformní celulární automaty, které porušují vlastnost homogenity. Formální definice takovéhoto automatu by mohla vypadat například takto: Opět uvažujme celulární automat jako množinu diskrétních hodnot Z^D v *D*-dimenzionálním prostoru. Protože lokální funkce každé buňky může být jiná, nelze použít jednu šablonu sousedství. Musíme tedy uvažovat pro každou buňku jinou možnou kombinaci sousedství. Formálně to tedy zapišme jako $\forall z \in Z^D \exists N \in F$ kde $|F| = |Z^D| = f$ a $F = \{N_0, N_1, \dots, N_{f-1}\}$. Dále pro *F* platí, že pokud zvolím libovolné dvě hodnoty $0 \leq i < f$ a $0 \leq e < f$, $e \neq i$ platí, že N_i a N_e můžou, ale nemusejí být stejné. Podobně zaveďme přechodové funkce jednotlivých buněk. Tento krok je nutný, protože musíme mít na paměti, že stejný počet sousedních prvků ještě neznamená použití stejných funkcí pro vyhodnocení výsledného stavu (toto tvrzení platí i obráceně). Zaveďme množinu přechodových funkcí $L = \{\Delta_0, \Delta_1, \dots, \Delta_{f-1}\}$, $|L| = f$. Dále existuje zobrazení $v: L \rightarrow F$, které přiřazuje platné šabloně sousedství platnou přechodovou funkci a zobrazení $w: Z^D \rightarrow L$, které dané buňce přiřadí její přechodovou funkci. Potom tedy přechod automatu z jednoho stavu v časovém okamžiku *t* do stavu *t+1* muže vypadat následovně.

$$s_{(t+1)}(z) = w(z)(v(z)) \qquad (2)$$

Možné rozšíření formální definice automatu tak, aby porušoval vlastnost lokality, by bylo následující. Uvažujme globální informaci jako přirozené číslo $g \in \mathbb{N}$. Je nutné modifikovat definici

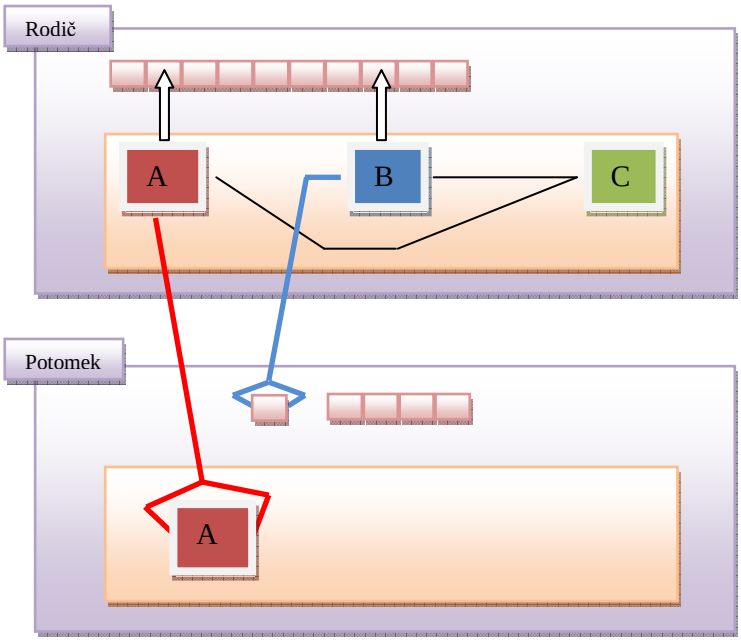
přechodové funkce jako $\Delta: Q^n \cup \mathbb{N} \rightarrow Q$. Přechod automatu z jednoho stavu v časovém okamžiku *t* do stavu *t+1* muže vypadat následovně.

$$s_{(t+1)}(z) = \Delta(s_t(z_0 + z), s_t(z_1 + z), \ldots, s_t(z_{n-1} + z), g) \qquad (3)$$
$$s_{(t+1)}(z) = w(z)(v(z), g)$$

2.3 Historie a von Neumannův celulární automat

Mezi nejvýznamnější průkopníky celulárních automatů patří zcela jistě John von Neumann. Jeho snaha směřovala k navržení stroje, který by byl schopen vytvářet své vlastní kopie. Zabýval se tedy sebereprodukcí. Své snahy publikoval již v roce 1948 na přednášce s názvem „Obecná teorie automatů“. Byl přitom inspirován prací W. McCullocha a W. Pittse. Ti se zabývali výzkumem umělých neuronových sítí. Samozřejmě čerpal i ze znalostí A. Turinga. Pokusil se tedy navrhnout automat, který je schopen sebereprodukce . Navrhl kinematický automat. Objevily se však komplikace s takzvanými black-box problémy [1]. V návrhu se totiž objevovaly součásti, které plnily jisté úkoly, ale jejich vnitřní struktura nebyla známa. S řešením přišel jeho spolupracovník a přítel Stanislaw Ulam. Jeho inspirace vycházela z růstu krystalů. Navrhl, že by prostředí mohlo být tvořeno jakousi šachovnicí, kde každé políčko tvoří jednu samostatnou buňku. Každá buňka může představovat konečný automat. Všechny automaty pracují se stejnou množinou pravidel. Zde již můžeme mluvit o logickém návrhu prvního celulárního automatu [2]. Tato myšlenka byla lákavá již z toho důvodu, že se snažila zvýšit rychlost počítačů pomocí paralelních výpočtů. Von Neumann na základě Ulamova schématu vytvořil první takovýto buněčný automat. Přirozené organismy jsou mnohem složitější než takovýto umělý automat. Na druhou stranu je možné pozorovat určité schody při pozorování těchto dvou systémů. Z tohoto důvodu von Neumann věřil, že je celulární automat vhodný pro simulaci a zkoumání primitivního života. Název celulárního automatu pochází od Arthura Burkse, který byl editorem von Neumannovych prací [1].

Von Neumannův automat byl sestaven přibližně z 200000 buněk. Každá z nich měla 29 stavů. Tělo automatu tvořilo asi 80x400 buněk. Tělo bylo rozděleno na tři složky a to továrnu, duplikátor a počítač. Další částí byl dlouhý výrůstek, který se skládal asi z 150000 buněk a byl analogií k pásce Turingova stroje [1]. V tomto automatu se již projevovala emergence. Ideu tohoto automatu zachycuje obrázek 2.



Obrázek 2.: Idea Von Neumannova automatu

Z obrázku 2 je dobře patrný princip tohoto zařízení. Blok označený písmenem *A* můžeme pojmenovat například jako továrnu. Stroj vysune rameno a začne replikace. Část označená symbolem *B* je nazvána jako duplikátor. Duplikátor přesune informaci z řídicí pásky rodiče do potomka. Blok označený *C* můžeme pojmenovat jako počítač z kinematického modelu. Celá činnost je řízena instrukcemi, které uchovává již zmíněná páska. Poté co je potomek dokončen, je oddělen od rodiče a může začít sám svoji vlastní sebereplikaci [2]. Sám von Neumann však nestihl vytvořit důkaz existence takovéhoto automatu. Ten byl publikován v roce 1964 Thatcherem.

Dá se tedy říci, že von Neumann a Ulam položili základy studií buněčných automatů. Po von Neumannově smrti se tímto směrem ubíralo několik učenců. Mezi ně patřil například A. Bruks a J. Holland. Holland navrhl flexibilní buněčný automat, který se dokázal přizpůsobit svému prostředí. Dokázal, že je možné celulární automat využít pro optimalizační úlohy. V neposlední řadě také položil základy genetickým algoritmům [3].

Díky odlišnosti tohoto směru od ostatních trendů tehdejší doby a neexistenci dostatečně výkonných výpočetních prostředků zájem o výzkum celulárních automatů a komplexity upadal. Z ústraní vynesl celulární automaty až J. H. Conway ze svoji hrou „Life“ [6]. Byl dobře obeznámen s prací svých předchůdců a navrhl pravidla pro struktury vyskytující se ve hře. Podařilo se mu docílit opravdu zajímavého komplexního chování. Výsledky Conwayovy práce byly představeny v časopise Scientific American. Díky hře Life se celulární automaty rozšířily do celého světa a staly se běžnou součástí vědeckého vnímání umělého života. Dnes se celulární automaty využívají nejen k simulaci umělého života, ale i v dalších oblastech [3].

Díky paralelismu, který buněčné automaty přináší, se objevily i snahy vytvořit hardwarovou podporu celulárních automatů. První takovéto pokusy se objevily v 50. letech jako kombinace televize a počítače. Dále vznikla architektura Cellular Automata Machine (CAM) na MIT [3].

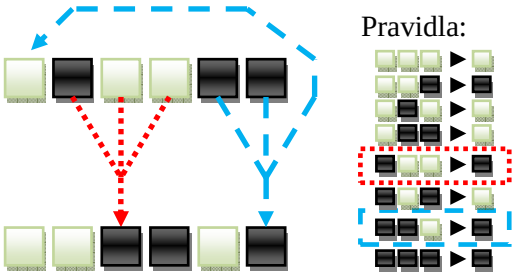
2.4 Členění celulárních automatů

Celulární automaty je možné členit podle mnoha kriterií. Jak již bylo uvedeno v kapitole 2.1, je možno dělit automaty podle stavů, kterých mohou jednotlivé buňky nabývat, podle druhu a velikosti sousedství, dle definic lokálních funkcí a podobně. Jedním z nejčastějších dělení, se kterým se při studiu těchto automatů setkáme, je podle prostoru, v němž jsou realizovány. Nejčastěji se setkáme s jednorozměrnými (1D) a dvojrozměrnými (2D) celulárními automaty.

2.4.1 1D celulární automaty

Jednorozměrným celulárním automatem máme na mysli takové seskupení, ve kterém má každá buňka maximálně dva přímé sousedy. Opět můžeme vyjmenovat nespočet druhů 1D celulárních automatů. Uvažujme tedy automat, jehož buňky nabývají pouze dvou stavů. Takovýto automat se nazývá binární. Uvažujme jistou množinu pravidel, podle které automat pracuje. Pokud necháme takovýto automat běžet a budeme zakreslovat pod sebe stavy jednotlivých buněk v po sobě jdoucích krocích, docílíme zajímavých výsledků. Výsledkem jsou různé obrazce a struktury, dle kterých můžeme zavést jedno z možných členění celulárních automatů.

S takovýmto automatem experimentoval Stephen Wolfram [1]. Výhodou 1D celulárních automatů je relativně malý počet jednoduchých pravidel a přehledná reprezentace výsledků. Sestavil tedy binární celulární automat. Nový stav vyšetřované buňky vycházel ze stavu této buňky a jejího levého a pravého souseda. Jedná se tedy o trojici hodnot. Tato trojce může nabývat osmi podob a těm lze přiřadit $2^8 = 256$ výstupních kombinací. Pro takovýto automat tedy existuje 256 možných pravidel [6]. Princip jednoho z těchto automatů zachycuje následující obrázek.



Obrázek 3.: Ukázka jednoho kroku 1D celulárního automatu a pravdivostní tabulky, podle které je řízen vývoj takového automatu.

Wolfram provedl experimenty s těmito automaty a na základě jistých společných vlastností výstupních struktur je rozdělil do čtyř tříd. Automaty, které jsou na následujících obrázcích uvedeny, startují vždy z náhodně generovaného počátečního stavu (řádku). Obrázky jsou převzaty z [6]. Popis tříd je vychází ze zdroje [1],[6].

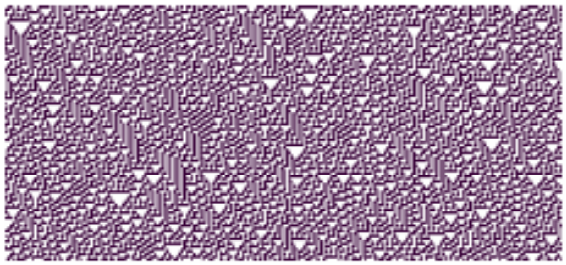
Třída I. – chování buněk automatu v této třídě dospěje po určitém počtu kroků do ustáleného stavu a již se nemění. Příklad je uveden na obrázku 4.

Obrázek 4.: I. Wolframova třída

Třída II. – struktury v této třídě se po výrazné počáteční aktivitě ustálí v jednoduchých stabilních shlucích nebo se v nich vyskytují jednoduché cyklicky se opakující útvary.

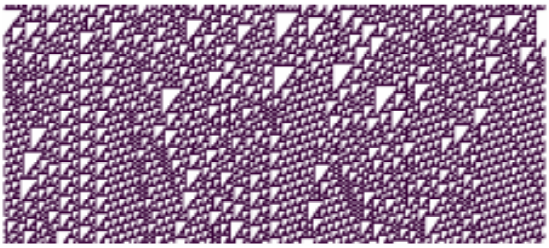
Obrázek 5.: II. Wolframova třída

Třída III. – vývoj působí chaoticky a pouhým okem není možné spatřit pravidelnost. Automaty z této třídy je možné využít ke generování šumu a pseudonáhodných čísel.



Obrázek 6.: III. Wolframova třída

Třída IV. – Tato třída produkuje složité, ale pravidelné struktury. Produkce je poměrně dlouhá. Do této třídy by bylo možno zahrnout i hru Life.



Obrázek 7.: IV. Wolframova třída

Další význačnou osobností, která přispěla k vývoji celulárních automatů, byl Langton. Ten je znám především díky Q-smyčkám, jejichž pravidla sestavil a kterým bude věnována jedna z dalších částí této práce (4.4). Langton však zavedl i kvantitativní hodnocení dynamiky těchto automatů. Toto hodnocení popisuje míru schopnosti přenosu informace. Langton zastával názor, že přenos a uchovávání informace jsou základní charakteristiky živých organismů [1]. K vyjádření schopnosti přenosu pravidel byl zaveden parametr λ [6]:

$$\lambda = \frac{(K^N - n)}{K^N},$$

kde N je počet sousedů s vyšetřovanou buňkou. K je počet možných stavů buňky a n vyjadřuje počet pravidel vedoucích ke klidovému stavu. $(K^N - n)$ je tedy počet pravidel, jejichž výstupy jsou neklidové stavy. K^N je počet všech pravidel. Neklidový stav je opakem stavu klidového, který můžeme definovat jako situaci, kdy má buňka, která je v klidovém stavu, ve svém okolí pouze buňky, které jsou také v klidovém stavu. Poté se stav této buňky v další generaci vývoje tohoto automatu nezmění.

Langton byl samozřejmě seznámen s Wolframovou klasifikací. Zavedl tedy kvantitativní hodnocení dynamiky těchto tříd [1],[6].

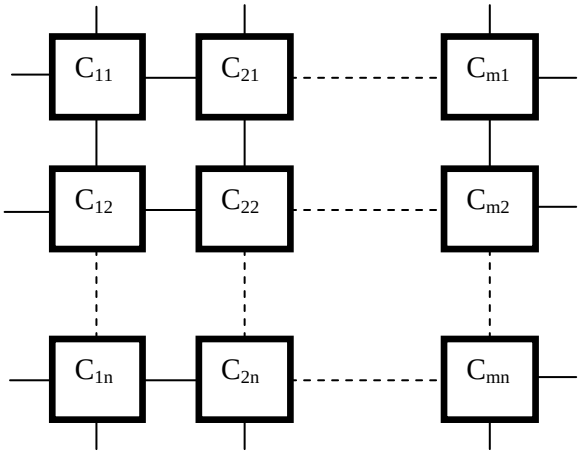
Malé hodnoty λ – do této kategorie patří I. a II. Wolframova třída. Informace je „zmražená“. To znamená, že je možné ji dlouho uchovávat, ale nelze ji přenášet.

Velké hodnoty λ (≈ 1) – do této kategorie patří III. Wolframova třída. Informace je na rozdíl od předcházející skupiny přenášena velmi lehce, často až chaoticky. Informaci však nelze uchovávat.

Mezní (střední) hodnoty λ ($\approx 0{,}5$) – do této kategorie patří IV. Wolframova třída. Přenos informace je teoreticky možný. Nedosahuje takových rychlostí, aby se ztratila vazba na její původní místo. Právě tato třetí skupina je podle Langtona vhodná pro simulaci života. Navazoval totiž na názory von Neumanna, že život existuje na samém pokraji chaosu a řádu.

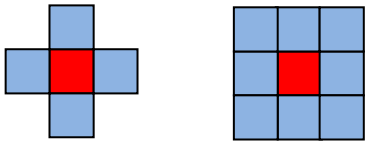
2.4.2 2D a vícerozměrné celulární automaty

Dvojrozměrné celulární automaty jsou velmi rozšířené a využívají se v mnoha aplikacích. U dvojrozměrných automatů skupina buněk tvoří mřížku, kterou si můžeme představit jako šachovnici a buňku samotnou jako čtverec. Příklad ukazuje obrázekObrázek 8.



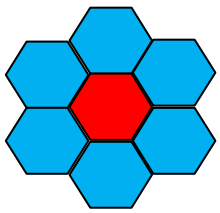
Obrázek 8.: Schéma 2D celulárního automatu

Teoreticky může být tento automat nekonečně velký. Prakticky je nutné uvažovat okrajové podmínky. Tyto automaty mají samozřejmě složitější přechodové funkce než 1D celulární automaty. Okolí buňky může být libovolné. V praxi se velmi často setkáváme s dvěma pojmy a to Neumanovské (obrázek 9 vlevo) a Moorovo (obrázek 9 vpravo) okolí [3]. Červená buňka znázorňuje vyšetřovanou buňku a modré buňky pak tvoří její sousedství.



Obrázek 9.: Neumanovo a Moorovo okolí

Je samozřejmé, že čím je rozsáhlejší okolí, tím je složitější přechodová funkce. Existují i celulární automaty, u kterých si můžeme buňku představit jako šestiúhelník. Takovýto příklad je na obrázku 10. 3D a více rozměrné celulární automaty jsou samozřejmě možné, ale nejsou tak rozšířené, protože se zvyšují nároky na výpočetní systém, který by chod takovéhoto automatu simuloval. Zároveň roste složitost a počet přechodových pravidel.



Obrázek 10.: Šestiúhelníkové okolí

2.4.3 Celulární automaty bez předpokladu pravidelné mřížky

Existují i celulární automaty širší koncepce, které nemají uspořádání pravidelné mřížky. Jedním z takovýchto systému je Reynoldsův model shlukování ptáků [1]. V roce 1987 Reynolds navrhl počítačový model chování hejna ptáků bez vůdčího jedince, který vycházel ze tří pravidel. Prvním pravidlem bylo shromažďování, kdy jedinec směřuje k těžišti svých sousedů. Dále pak synchronizace rychlosti, kdy se snaží jedinec synchronizovat svoji rychlost s rychlostí svých sousedů. Posledním pravidlem bylo vyhýbání se kolizím. Pokud se jedinec neúměrně přiblížil k překážce, snažil se změnil okamžitě svoji polohu. Tento model byl inspirován hejnem kosů, které Craig Reynolds pozoroval. Virtuální jedinci v modelu se nazývají *boidové*. Když svoji simulaci předvedl na SIGGRAPH (konference zabývající se počítačovou problematikou a vývojem v tomto odvětví), ornitologové přijali hypotézu, že chování hejna vyplývá ze spolupráce rovnocenných jedinců podle několika jednoduchých pravidel bez nutnosti řídicí autority. Jeho model byl dokonce použit v animacích k některým filmům (například Batman se vrací).

Podobné experimenty prováděl i Langton (chování mravenců) a Deneubourg (chování termitů). Oba dospěli k podobným výsledkům jako Reynolds. Tyto poznatky byly odsouhlaseny i entomology.

2.4.4 Dynamické celulární automaty

Tyto automaty, na rozdíl od všech předchozích, mohou měnit počty svých buněk. Přechodová funkce neříká pouze jaký stav má buňka v dalším kroku získat, ale může obsahovat informaci, na kolik buněk a s jakými počátečními stavy, se má buňka rozštěpit. Ve spojitosti s 1D celulárními automaty mluvíme o takzvaných L-systémech [1]. Ty definoval Aristid Lindenmayer a můžeme je popsat modifikací formální gramatiky. Byly určeny pro modelování vývoje mnohobuněčných organismů. Pravidla jsou v zásadě používána rekurzivně, to zapříčiňuje soběpodobnost, což je charakteristická vlastnost fraktálu. I jednoduchá soustava pravidel je tedy schopna simulovat proces růstu složitých struktur.

2.4.5 Celulární automaty s vysokým počtem stavů

Dalším kritériem, jak můžeme posuzovat celulární automaty, je počet stavů, kterých může buňka nabývat. Buňky automatů zmiňovaných v předcházejících kapitolách nabývaly pouze několika málo stavů. Existují však i automaty pracující s enormně vysokým počtem stavů. Jedním z nich je Arbibův celulární automat [1]. Pracuje s von Neumannovým okolím s 2^{279} stavy. Každá ze sousedních buněk je napojena na vstupně výstupní datový kanál a na vazební člen. Instrukční sada obsahuje sedm instrukcí. Obsahuje registry pro vnitřní program a pro vazební členy. Stavy těchto součástí jsou východiskem pro další stavy. V některých případech dochází k pohybům skupin buněk svázaných zmiňovanými vazebními členy.

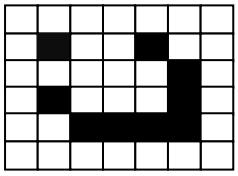
2.5 Hra Life

Jedna z nejznámějších a nejrozšířenějších aplikací využívajících celulární automaty je dílem matematika cambridgské university Johna Hortona Conwaye [3]. Název hra Live je poněkud zavádějící, protože ji hráč vlastně nehraje. Neposkytuje do ní žádné vstupy (tedy kromě inicializace a případného přidávání entit). Inspiroval se prací von Neumanna. Vycházel z 2D celulárních automatů, které mají pouze dva stavy. Buňka buďto obsahuje život, nebo je prázdná. Buňka zná svůj stav a stavy buněk v bezprostředním okolí.

Chování hry je dáno jednoduchými lokálními pravidly [3]. Pro živou buňku platí, že jestliže má kolem sebe méně jak dvě živé buňky, tak umírá na osamělost. Pokud ale má kolem sebe více jak tři buňky, umírá na přesycení. Tedy tyto zákonitosti se dají shrnout tak, že pokud má buňka kolem sebe dvě nebo tři živé buňky, pak přežívá. Naopak, pro mrtvou buňku platí, že pokud má kolem sebe právě tři živé buňky, tak ožívá. Aplikací těchto pravidel v inicializovaném celulárním automatu docílíme zajímavého chovaní. Můžeme pozorovat různé struktury, které můžeme rozdělit do několika skupin.

- struktury, které po několika krocích (generacích) zaniknou,
- cyklicky se opakující struktury, setrvávající na stejných pozicích,
- cyklicky se opakující struktury, které se posouvají,
- stabilní struktury, které nezaniknou.

Conway chtěl ukázat, že je jeho jednoduchý celulární automat universum, do kterého je možné vsadit Turingův stroj, tedy sestrojit z jeho automatu počítač [1]. Bylo tedy nutné nalézt obrazec, který by přenášel informaci (který by se pohyboval) a obrazec, který by tyto nosiče produkoval. Tento obrazec nalezl William Gosper z MIT. Jednalo se o takzvané „kluzákové dělo“, které produkovalo každých třicet generací strukturu nazvanou „kluzák“ (glider, Obrázek 11). Ten se pohyboval čtvrtinou „rychlosti světla“, tedy jedno políčko po diagonále ve čtyřech generacích. Byly objeveny i další pohyblivé struktury. Také bylo dokázalo, že se žádná z nich nemůže pohybovat „rychlostí světla“ (jedno políčko v kterémkoli směru v jedné generaci). Celý počítač však nikdy v Life nebyl sestrojen, pouze fungující sčítačka. Nikdy se také nepodařilo vytvořit konečnou sebereplikující se strukturu (vytvářející své kopie, aniž by sama zmizela). Odhaduje se však, že na mřížce 10^6 na 10^6 políček lze vytvořit strukturu rovnocenou s jednobuněčným živým organismem [1].



Obrázek 11.: Kluzák (glider)

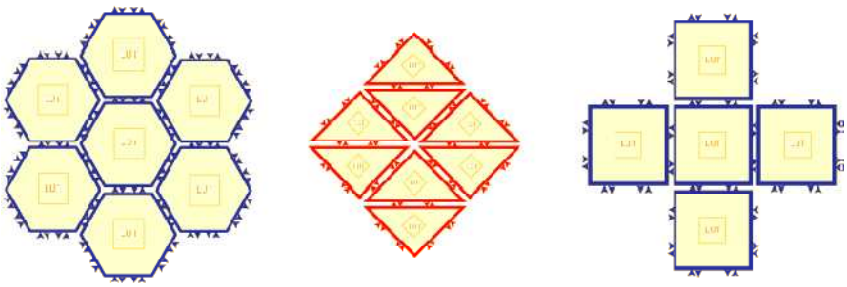
3 Architektura Cell Matrix

Jedná se o architekturu založenou na principu celulárních automatů. Cell Matrix má obecné uplatnění. Vykazuje známky masivního paralelismu a je parciálně a distribuovaně rekonfigurovatelný [7]. Je sestaven z matice jednoduchých buněk. Ty jsou vzájemně propojeny na lokální úrovni. Buňky jsou homogenní. Typy sousedství, které buňka realizuje, závisí na konfiguraci a konstrukci buňky. Každá z buněk obsahuje malou paměť. Je schopná realizovat 8 logických funkcí o čtyřech vstupech nebo jednoduché kombinační obvody jako jsou jednobitová úplná sčítačka, klopné obvody a další. Uspořádání buněk muže být ve 2D nebo 3D mřížce. Cell Matrix je zajímavé prostředí, ve kterém je možno realizovat samoupravující se systémy, seberekonfiguraci a sebereplikaci [7].

Zajímavým rysem této architektury je, že každá z buněk je jak aktivním prvkem struktury, tak i objektem, který může konfigurovat ostatní prvky a sám muže být ostatními buňkami konfigurován. Díky tomu může docházet k rekonfiguraci obvodu na více místech současně.

Buňky této technologie se mohou vyrábět v několika variantách, jak je znázorněno na obrázku 12, ten je převzat ze [7]. V další části této práce se zaměřím na buňky čtvercové (uspořádané do pravidelné 2D mřížky).

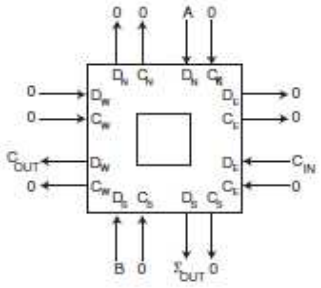
Buňky mohou pracovat v jednom ze dvou módů. První z nich je datový a nazývá se D-mód. Druhý je řídicí a označujeme jej jako C-mód.



Obrázek 12.: Varianty buněk v Cell Matrix

3.1 D-mód buňky

Datový mód umožňuje buňce pracovat jako kombinační obvod. Pokud budu uvažovat čtvercovou buňku ve 2D uspořádaní, vypadají její vstupy jako na obrázku 13, který je převzat z [4].



Obrázek 13.: Vstupy buňky

Abychom nastavili buňku do D-módu, musíme na všechny její vstupy C_{IN} přivést hodnotu log. 0. Pokud je buňka nastavena do tohoto režimu, můžeme pomocí vstupů $D_{IN}(N,E,S,W)$ nastavit 4 bitovou adresu do paměti buňky. Paměť buňky je definována pravdivostní tabulkou o velikosti 8x16 bitů. Hodnoty z tabulky (8 bitů) se pak přivedou na výstupy $D_{OUT}(N,E,S,W)$ a $C_{OUT}(N,E,S,W)$. Příklad tabulky je znázorněn na obrázku 14, který je převzat ze [4]. Tento mód je asynchronní, tudíž není řízen hodinovým signálem. Buňky se chovají jako neuniformní celulární automat. Více je možné najít ve zdroji [4],[7].

Inputs				Outputs							
D_{IN} N	D_{IN} S	D_{IN} W	D_{IN} E	C_{OUT} N	C_{OUT} S	C_{OUT} W	C_{OUT} E	D_{OUT} N	D_{OUT} S	D_{OUT} W	D_{OUT} E
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	0	0	1	0
0	0	1	1	0	0	0	0	0	1	1	0
0	1	0	0	0	0	0	0	0	1	0	0
0	1	0	1	0	0	0	0	0	0	1	0
0	1	1	0	0	0	0	0	0	1	0	0
0	1	1	1	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	1	0	0
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	0	0	1	0
1	1	0	0	0	0	0	0	0	1	0	0
1	1	0	1	0	0	0	0	0	1	1	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	1	1	0

Obrázek 14.: Tabulka buňky Cell Matrix

3.2 C-mód buňky

Konfigurační mód buňky umožňuje změnu hodnot v pravdivostní tabulce. Na konfiguraci pravdivostní tabulky se podílejí všechny sousední buňky, které sdílí aktivní stranu buňky v C-modu. Aktivní strana je taková, která má vstup C_{IN} nastaven na hodnotu log.1. V tomto módu funguje paměť buňky jako posuvný registr. Celý proces je řízen hodinovým signálem. Při nástupné hraně je vypočtena nová hodnota jako logický součet (OR) všech D_{IN} aktivních stran. Při sestupné hraně je pak nově vypočtený bit zapsán na první místo tabulky. Ostatní hodnoty jsou posunuty (posuvný registr) a hodnota posledního bitu je odeslána na D_{OUT} aktivních stran. Na všech C_{OUT} (jak aktivních, tak neaktivních stran) a D_{OUT} neaktivních stran je vždy zapsána log.0. Tento systém umožňuje libovolnou konfiguraci pravdivostní tabulky. Více je možné najít ve zdroji [4],[7].

Tato vlastnost také umožňuje distribuovanou rekonfiguraci. Je tedy možné konfigurovat i vzdálenou buňku. Nejprve nakonfigurujeme pravdivostní tabulky buněk mezi dvěma zvolenými buňkami do režimu podobnému routeru (tedy propouští komunikaci dále) a poté konfigurujeme pravdivostní tabulku vyšetřované buňky.

4 Sebereplikace v celulárních systémech

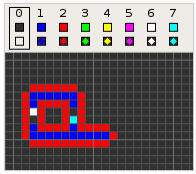
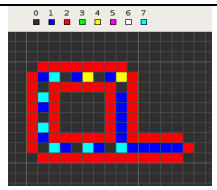
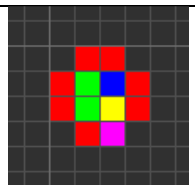
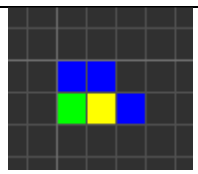
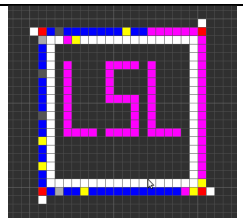
Jak již bylo popsáno v úvodní kapitole této práce, sebereplikace by se dala využít například k vytvoření armády robotických pracovníků. To není zdaleka jediné využití. Dalším příkladem mohou být nanoboti, kteří se množí v našem těle a bojují proti různým virům a nemocem. V neposlední řadě by bylo možné využít tyto systémy pro vojenské účely. To je však spíše vizí budoucnosti. Jak je tato budoucnost vzdálená, již záleží na pokroku, který věda v tomto směru udělá.

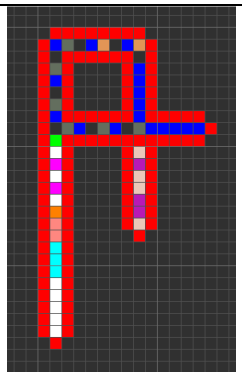
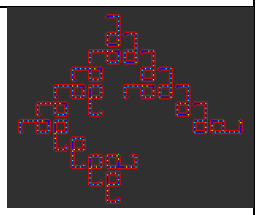
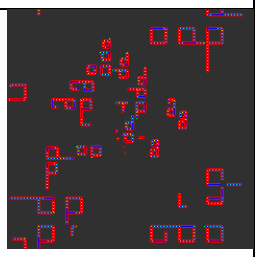
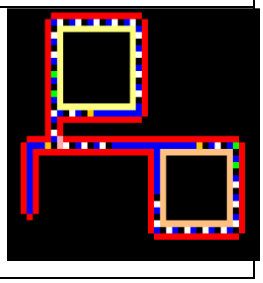
V současné době je s termínem sebereplikace spíše spojováno množení se datových struktur. V dalších částech této práce se zaměřím na sebereplikace v celulárních systémech. Inspirací těmto myšlenkám je z části i skutečný život. I mikroorganismy se rozmnožují. Vytvářejí tak své vlastní repliky. Jak je ale možné tento proces simulovat v počítačích? Základní teze umělého života je, že podstatou života je informace a ne materiální forma. Tato materiální forma je pouze obálkou pro uchování informace a samotné rozmnožování je pak procesem předávání informace [1]. Samotný život však vyžaduje jistou míru složitosti. Po dosažení této složitosti jsou struktury schopny rozmnožování se. V jistém případě můžeme dosáhnout i toho, že potomci nebudou jen kopií svého rodiče, ale budou dokonalejší a složitější. Prostředkem vývoje dokonalejších jedinců může být například evoluce, která využívá mutace, selekce a sebereprodukce. Evoluce simulována v počítači může nabývat mnoha rozměrů. Nemusíme předpokládat pouze klasickou darwinovskou teorii, ale například i lamarckistickou. Ta říká, že prostřednictvím předávání získaných vlastností je možné dosáhnout vyšší dokonalosti. V úvahu je možné brát i Baldwinův efekt. Ten říká, že to, co se jedinec během své existence naučí, nepředá přímo svým potomkům, ale zvyšuje to jeho šance na rozmnožování a tím i pravděpodobnost výskytu jeho genů v další generaci. Všechny tyto vlastnosti mohou být klíčové pro vývoj umělých sebereplikujících se struktur. Tento odstavec vycházel ze zdroje [1].

V celulárních systémech je vhodnou volbou lokálních pravidel a počáteční konfigurací možno dosáhnout struktur, které se po určitém počtu generací (kroků) rozmnoží. Důležitou otázkou je však to, jestli je možná sebereplikace libovolné struktury, která by přenášela netriviální a užitečnou informaci nebo prováděla univerzální výpočet. Již von Neumann ukázal, že je to možné (více v kapitole 2.3). V jeho stopách pak pokračovali další odborníci, z nichž je zřejmě nejznámější Langton [2].

4.1 Přehled sebereplikujících se struktur

Následující tabulka (Tabulka 1) zobrazuje přehled nejznámějších struktur schopných sebereplikace. Jednotlivé struktury budou podrobněji popsány v dalších kapitolách práce. Podklady pro tabulku jsou převzaty z [10]. Údaje v tabulce jsou řazeny chronologicky podle data vzniku. Datových struktur samozřejmě existuje mnohem více, ale v této práci se zaměřím pouze na níže uvedené struktury.

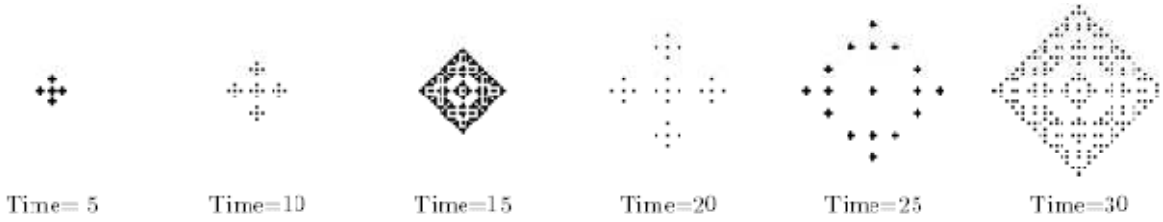
Celulární struktura	**Počet stavů**	**Uvažované okolí**	**Typický počet buněk**	**Typický počet kroků potřebných pro sebereplikaci**	**Ukázka**
Jednoduché binární struktury	2	Von Neumann/ Moor	4/8	různé	
Coddův automat (1968)	8	Von Neumann	různé	různé	
Langtonovy Q-smyčky (1984)	8	von Neumann	86	151	
Bylovy smyčky (1989)	6	von Neumann	12	25	
Chou-Reggia smyčky (1993)	8	von Neumann	5	15	
Tempesti smyčky (1995)	10	Moor	148	304	

Perrierovy smyčky (1996)	64	von Neumann	158	235	
SDSR smyčky (1998)	8	von Neumann	86	151	
Evoloop (1999)	8	von Neumann	149	363(v závislosti na evoluci a velikosti jedince)	
Sexyloop a F-Sexyloop (2006-2008)	10 a 12	Von Neumann	různé	různé	

Tabulka 1.: Přehled sebereplikujících se struktur

4.2 Jednoduché sebereplikující se struktury, XOR pravidlo

V této i v následujících kapitolách uvažujme dvojrozměrný uniformní celulární automat. Prvním příkladem sebereplikujících se struktury může být struktura zachycená na obrázku 15, který je převzat z [2].



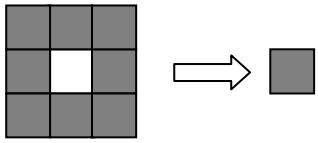
Obrázek 15.: Jednoduchá sebereplikující se struktura, XOR pravidlo

Buňky automatu realizujícího tuto strukturu mohou nabývat dvou stavů (pro korespondenci s obrázkem si je označme jako černý a bílý stav). Z obrázku je patrné, že po určitém počtu kroků (generací) je dosaženo mnohonásobné replikace struktury. Jedná se o automat řízený pravidlem XOR. V potaz se bere von Neumannovo okolí. Pravidlo definujme následujícím vztahem. Definice je převzata ze zdroje [2].

$$s_{(t+1)}(z) = XOR_{i=0}^{n-1} s_t(z + z_i) = \left(\sum_{i=0}^{n-1} s_t(z + z_i)\right) mod\ 2 \qquad (4)$$

Podobnou strukturu, která provede svoji sebeaplikaci, znázorňuje i obrázek 16, který je převzatý z [6]. Zde již po čtyřech generacích (krocích) dojde k rozštěpení struktury na čtyři totožné objekty. Zde se uplatňuje Moorovo okolí.

Obrázek 16.: Samoreprodukující se struktura



Obrázek 17.: Příklad pravidla používající Moorovo okolí

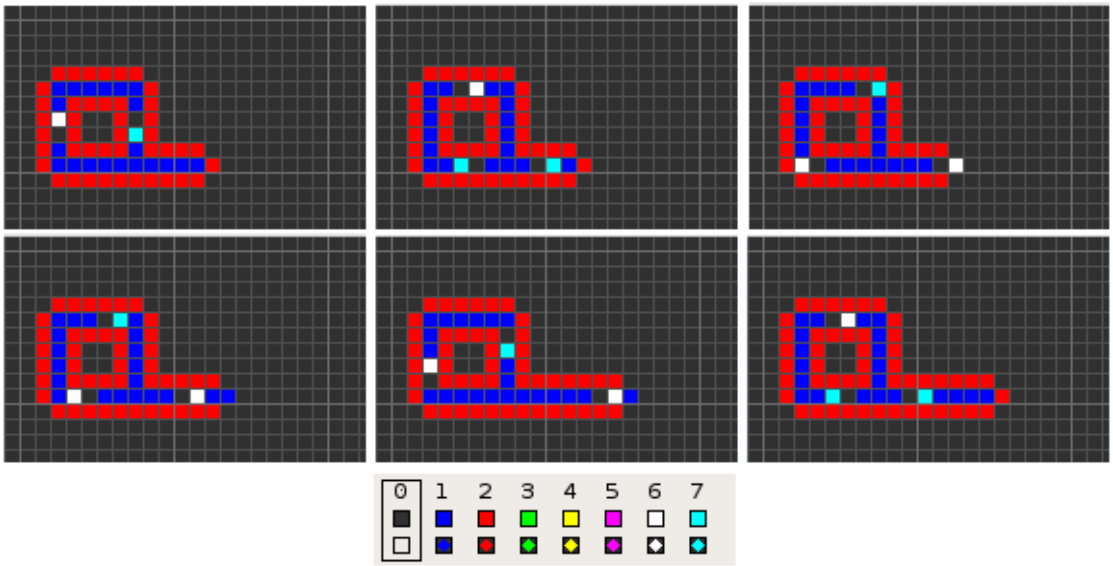
Z definice umělého života, kterou jsem uvedl v předchozí kapitole, vyplývá, že by tyto struktury měly dosahovat určitého stupně složitosti, aby mohli přenášet jistou užitečnou informaci.

Struktury uvedené výše, v této podobě, tuto vlastnost nesplňují, a tudíž se na ně dívejme pouze jako na prvotní příklad sebeaplikací v celulárních systémech.

4.3 Coddův automat

Tento automat se stal inspirací pro Langtonovy Q-smyčky. Vytvořil jej v roce 1968 F. G. Codd. Coddův automat pracuje s osmi stavy a von Neumannovým okolím [1]. Čtyři stavy jsou strukturální. Označují prázdnou buňku, signálovou cestu, obal signálové cesty a stav pro speciální použití. Zbylé čtyři stavy byly signálové. Základní signálový prvek zde tvoří dvojce signálové buňky v kombinaci s prázdnou buňkou. Tento signál se v generaci posune o jednu pozici po signálové cestě. Používá kolem 500 pravidel z 32768 možných. Tento automat je významný i v tom, že byl teoreticky schopen emulace Turingova stroje, který by byl schopen vytvořit svojí vlastní kopii. V tomto pohledu se tedy jedná o samoreprodukující se celulární automat.

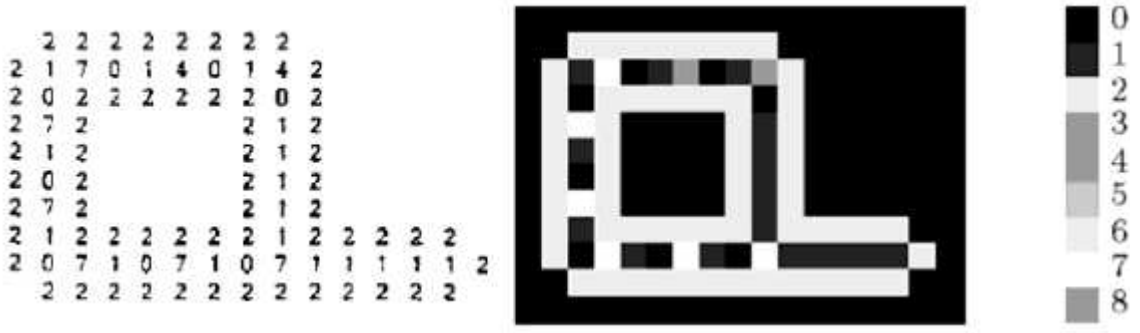
Následující snímky (Obrázek 18) zobrazují princip a částečnou funkci tohoto automatu. Jedná se o jednoduchou konfiguraci. Dva signály obíhají kolem smyčky a střídavě se duplikují. Nejprve je generován signál „07“, který rozbijí rameno. Poté je generován signál „06“, který rameno opraví. Společně prodlouží tyto signály rameno o jednu buňku. Tento proces se opakuje. Kompletní signál pro prodloužení ramena o jednu buňku by tedy byl „70116011“. Signály je třeba oddělit dvěma stavy „1“, aby nedocházelo k rušení. Například prodloužení se zahnutím doleva by mělo tento tvar „4011401150116011“. Obrázek 18 je vytvořen pomocí zdrojů z [13], kde je možné najít podrobnější popis problematiky.



Obrázek 18.: Funkce Coddova automatu

4.4 Langtonovy Q-smyčky (SR-Loops)

Christofer Lagton sestrojil v roce 1984 verzi mnohem jednoduššího samoreprodukujícího se 2D celulárního automatu než byl ten Coddův [1]. Neopíral se již o myšlenku emulace Turingova stroje. Využíval 219 pravidel využívající von Neumannovo okolím. Název Q-smyčky vychází z tvaru struktury. Velmi často se označují jako SR-Loops (Self Reproducing loops) [2].



Obrázek 19.: Langtonova sebereplikující se smyčka

Obrázek 19 ukazuje základní tvar této smyčky. Tento obrázek je převzat z [6]. Jak je patrné, smyčka využívá osmi stavů. Jednotlivé stavy můžeme rozdělit do dvou základních skupin. První z nich jsou takzvané „základní stavy“. Druhou skupinou jsou „signály“. Podrobnější rozdělení zachycují následující tabulky (Tabulka 2, Tabulka 3), které vychází ze zdroje [2].

Stav	Jméno stavu	Funkce stavu
„0“	Pozadí	Pozadí, klidový stav celulárního automatu.
„1“	Jádro	Naplňuje trubici a vede v ní signál.
„2“	Pouzdro	Vytváří obal trubice.

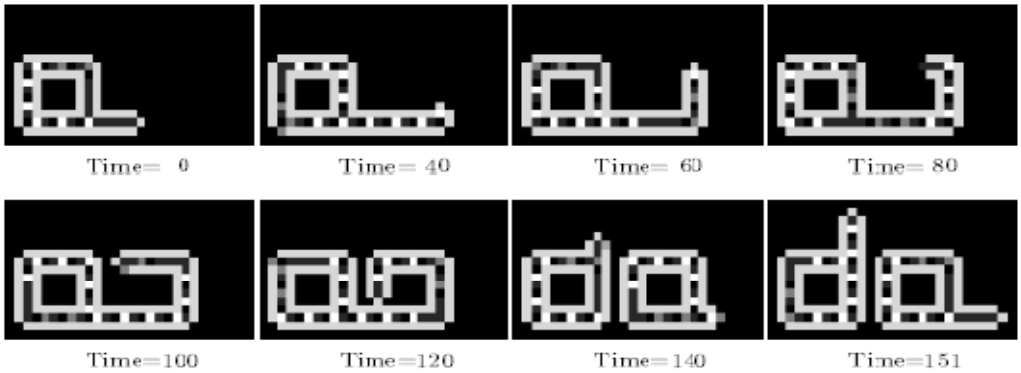
Tabulka 2.: Tabulka základních stavů

Stav	Jméno stavu	Funkce stavu
„3“	Levý indikátor, vázací prvek, stav pro podporu růstu	Podporuje natáčení ramene. Podporuje provázání dvou ramen. Podpora růstu nového potomstva.
„4“	Gen 4	Drží genetickou informaci o natáčení ramene.
„5“	Přerušovač pupeční šňůry, posel, podpora růstu	Přeruší pupeční šňůru mezi potomkem a rodičem. Bod, ve kterém by měl klíčit nový růst. Zajišťuje podporu růstu.
„6“	Posel, podpora růstu	Bod, ve kterém by měl klíčit nový růst. Zajišťuje podporu růstu. Ukončuje růst potomstva.

„7“	Gen 7	Zajišťuje rovný růst pupeční šňůry. Podpora růstu.

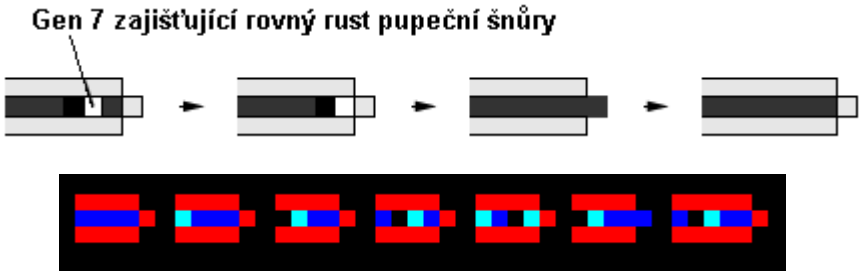
Tabulka 3.: Tabulka stavů

Jak je patrné, rozmnožování probíhá pomocí částí, kterou můžeme nazvat „pupeční šňůrou“. Pupeční šňůra se začne prodlužovat a natáčet do levé strany (proti směru hodinových ručiček). Začne se tedy uzavírat nová smyčka. Po uzavření této smyčky dochází k rozpuštění pouta (pupeční šňůry) a začínají existovat dvě nezávislé smyčky. Tento proces je zachycen na obrázku 20, ten je převzat ze zdroje [2].



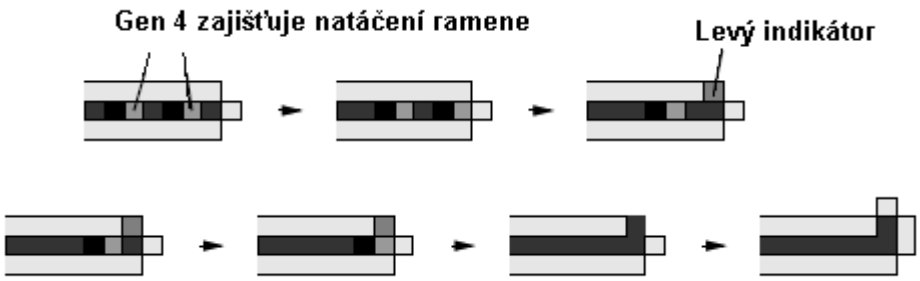
Obrázek 20.: Replikace Langtonovy smyčky

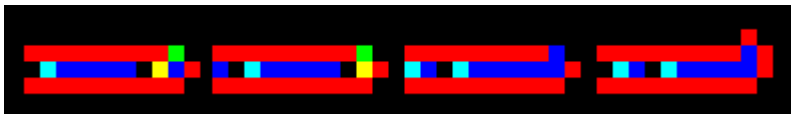
V obrázku 20 jsou jasně patrné již zmiňované kroky. V časovém intervalu 0-40 dochází k onomu prodloužení pupeční šňůry. Tento rovný růst zajišťuje stav „7“. Obrázky níže jsou převzaty z [2]. Ilustrace s tmavým pozadím jsou snímky z prováděné simulace. Následující obrázky 21 až 24 znázorňují detailněji klíčové situace replikace.



Obrázek 21.: Růst nové smyčky

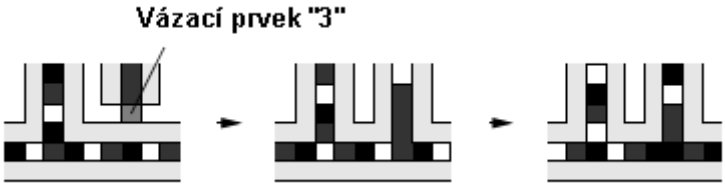
Poté dochází k zatáčení rostoucí pupeční šňůry do levé strany (proti směru hodinových ručiček). Toto točení zajišťují stavy „4“ a „3“.

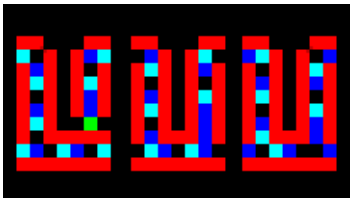




Obrázek 22.: Zatáčení pupeční šňůry

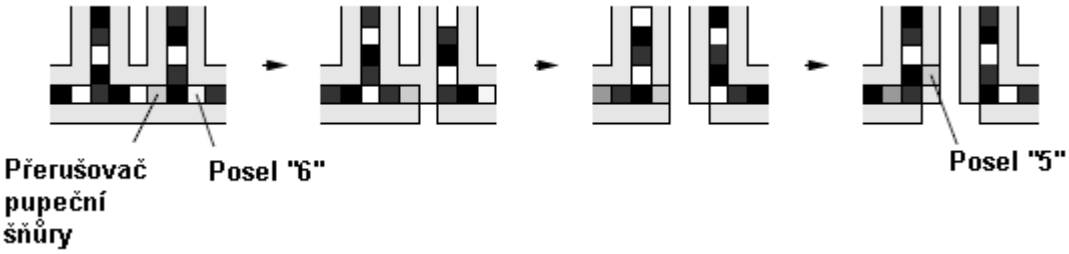
Po 120. časovém kroku dojde k propojení nové smyčky.

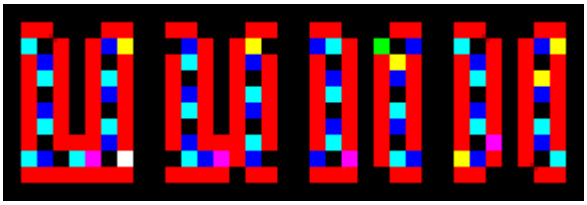




Obrázek 23.: Spojení smyčky

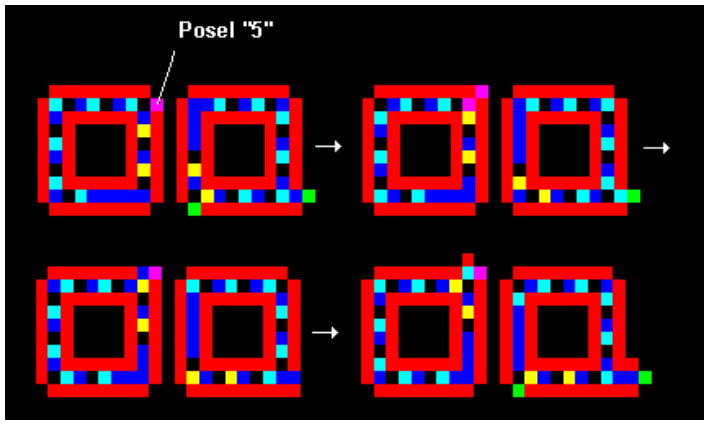
V dalších krocích dochází k rozpouštění pupeční šňůry. Smyčky pak pokračují v další reprodukci.





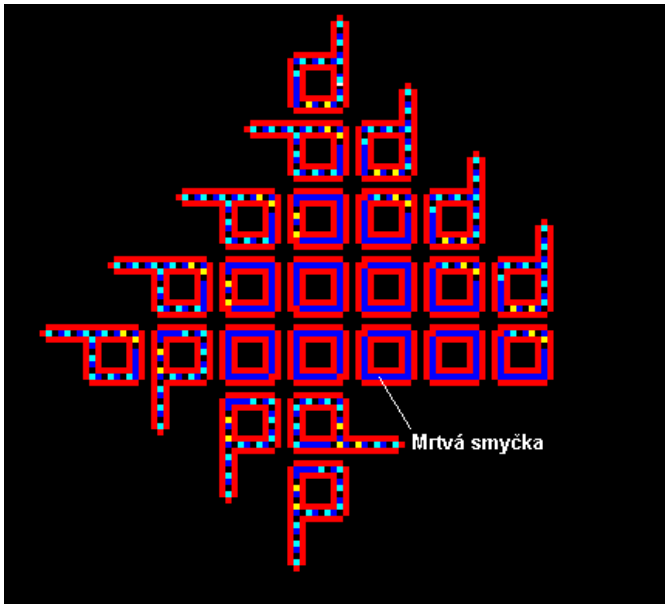
Obrázek 24.: Odpojení smyčky

Stav označený jako „5“ doputuje po straně smyčky na její okraj a pokud je prostor pro nového potomka, začne prodlužovat pupeční šňůru do tohoto prostoru.



Obrázek 25.: Replikace v dalších směrech

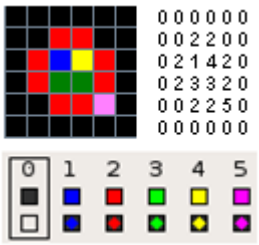
Tyto smyčky jsou názornou ukázkou existence dvojí funkce informace. Genotyp se kopíruje do potomka a fenotyp charakterizuje tvar a chování jedince. V tomto případě řídí tvorbu potomka. Natáčení je vždy proti směru hodinových ručiček. Pokud smyčka nemá prostor, kam by byla možná její replikace, můžeme říct, že umírá a zůstává po ní pouze torzo, které je nečinné. Takovéto smyčky je možno vidět na obrázku 26.



Obrázek 26.: Replikace Langtonových smyček

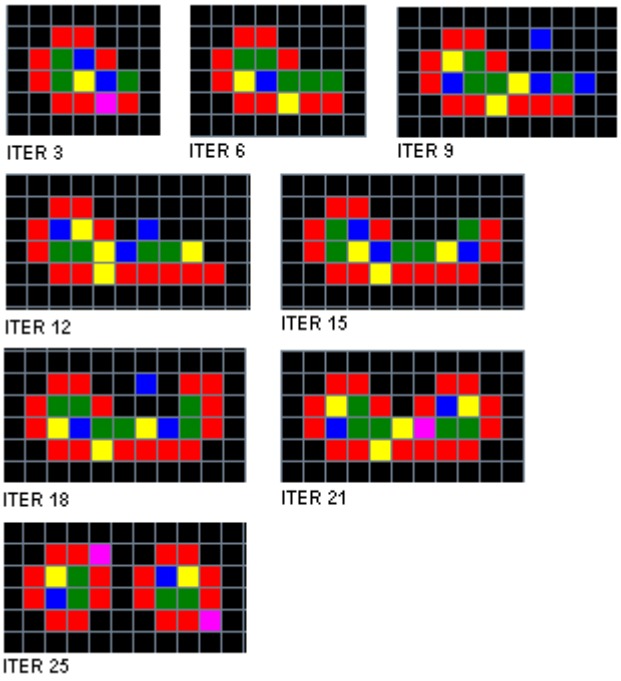
4.5 Bylovy smyčky (Byl's loop)

Několik let později, po Langtonovi (v roce 1989), zjednodušil John Byl jeho automat. Tento automat dokončí svoji sebereplikaci ve 25 krocích. Tato struktura má 5 stavů a zabírá 13 buněk.

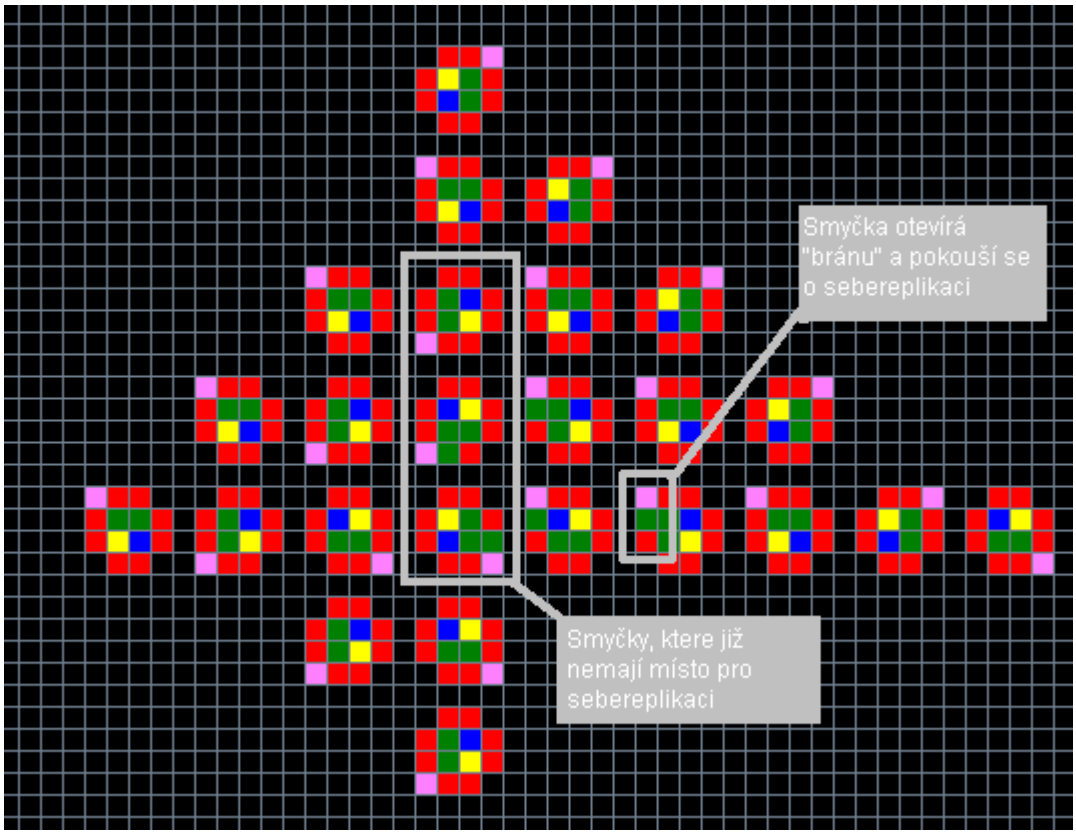


Obrázek 27.: Bylova smyčka

Bylova smyčka se stala základem pro zbytek mé práce a v dalších částech ji bude věnována velká pozornost. Je to díky stylu sebereplikace. Sebereplikace je znázorněna na následujících snímcích (Obrázek 28). Je podobná Langtonově smyčce. Stav „5“ společně se stavem „2“ (nad ním) tvoří takzvanou „bránu”, která se otevře. Začne se vysouvat rameno (Obrázek 28, INER 3-6). Poté co je rameno dostatečně vytaženo, dochází k růstu nové smyčky (Obrázek 28, INER 6-21). V další fázi je nová smyčka téměř dokončena. Rameno se začne stahovat zpět do původní smyčky a celá smyčka se natočí proti směru hodinových ručiček, aby mohla pokračovat v sebereplikaci v dalších směrech (Obrázek 28, INER 21-25). Více o této smyčce je uvedeno ve zdroji [16].



Obrázek 28.: Sebereplikace Bylovy smyčky

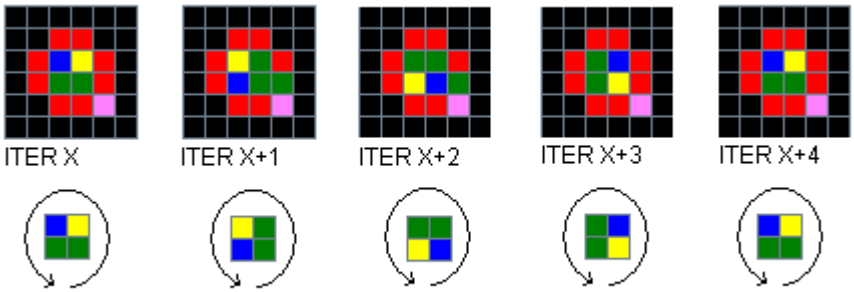


Obrázek 29.: Bylovy smyčky a jejich komunita

Na obrázku 29 je pozorovatelná velmi zajímavá skutečnost. Jedná se o fakt, že i po té, co smyčka nemá kolem sebe dostatek prostoru pro sebereplikaci, tak „neumírá“. Jádro smyčky neustále rotuje a při každé rotaci otevře bránu a pokouší se o sebereplikaci. Když zjistí, že nemá dostatek prostoru k replikaci, stáhne se a bránu opět uzavře. Tento fakt by mohl být v budoucnu využit například pro opravu poškozených smyček, nebo pro rozšíření vzájemné komunikace mezi jednotlivými smyčkami. Nesmrtelnost buněk také přispívá k teorii, že by smyčka mohla provádět užitečné výpočty.

V jistém slova smyslu tato smyčka již kromě sebereplikace jistou užitečnou činnost provádí. Jádro této smyčky neustále rotuje. Uvažujme například, že smyčka je motorem a její jádro je hřídelí, jenž má roztáčet generátor elektřiny. V tomto případě se tedy smyčky množí a po dokončení replikací fungují dále jako pohonné jednotky. Tato smyčka je o to zajímavější, že tato rotace probíhá již ve fázi sebereplikace, tudíž by jí bylo možno využít jako pohon hned od počátku bez přerušení. Po dokončení replikací navíc smyčky kontrolují, zdali se neuvolnilo kolem nich místo a není li možná další replikace.

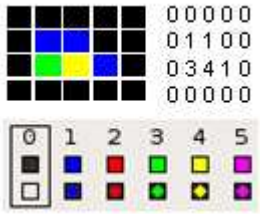
Následující obrázek 30 zachycuje rotaci jádra poté, co smyčka nemá kolem sebe dostatek prostoru k replikaci. Simulace dospěje do kroku X, kdy již kolem sebe smyčka nemá prostor pro replikaci, v kroku X+1 se otevírá brána a kontroluje se prostor. V kroku X+3 je brána uzavřena a od kroku X+4 se vše opět periodicky opakuje. Jádro smyčky neustále rotuje v protisměru hodinových ručiček.



Obrázek 30.: Rotace jádra Bylovy smyčky

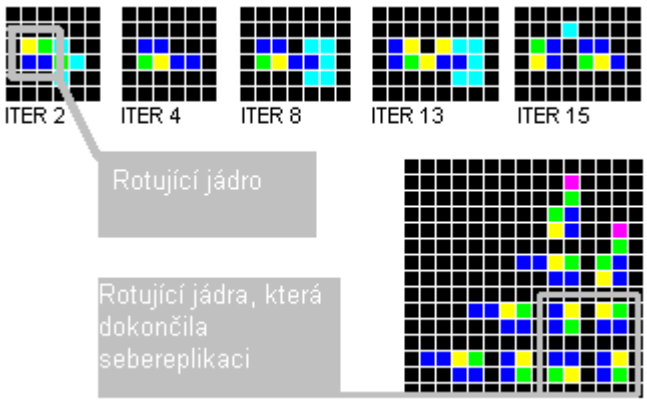
4.6 Chou-Reggia smyčka

J. A. Reggia a H. Chou sestavili v roce 1993 jednoduchou datovou strukturu, která má typicky pět buněk a osm stavů. Replikace probíhá v 15 krocích.



Obrázek 31.: Chou-Reggia smyčka

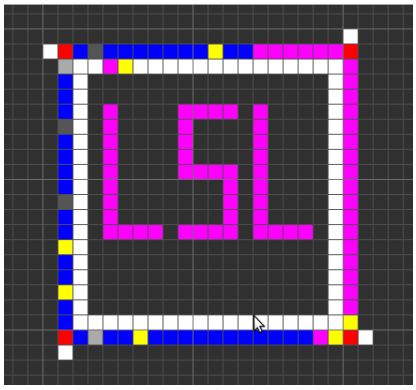
Tato smyčka je zjednodušení Langtonovy a Bylovy smyčky. Jedná se vlastně o samotné jádro Bylovy smyčky, které se replikuje. Po dokončení replikací zůstává pouze rotující jádro. Na následujících snímcích (obrázek 32) je znázorněna sebereplikace a dále komunita těchto struktur. Více informací je možno najít v [17].



Obrázek 32.: Replikace Chou-Reggia smyčky

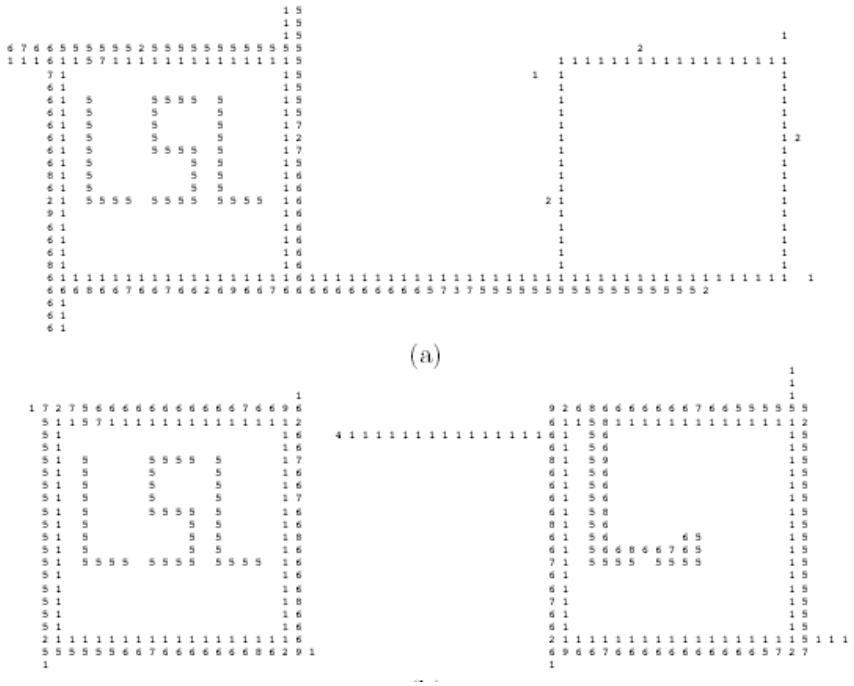
4.7 Tempesti smyčky

Struktury uvedené v předcházejících částech jsou sice velmi zajímavé a v určitém úhlu pohledu velmi fascinující, ale můžeme říci, že se jedná jen o polovinu příběhu. Tím je míněno, aby tyto smyčky, které vytváří sebereplikaci, mohli plnohodnotně sloužit lidem, musí také provádět užitečné výpočty. Langtonovy smyčky a jejich modifikace od tohoto účelu ustupují a zaměřují se na sebereplikaci (neuvažujme nyní rotaci jádra Bylovy a Chou-Reggia smyčky jako užitečný výpočet). G. Tempesti v roce 1995 vytvořil smyčku, která krom sebereplikace provádí i jistý užitečný výpočet. Tato struktura se replikuje a vypisuje do svého nitra nápis LSL (Logic Systems Laboratory). Automat však musí uvažovat Moorovo okolí, čímž se značně zvětšuje stavový prostor. Následující obrázek 33 ukazuje tuto smyčku. Obrázek 33 je převzat z [10].



Obrázek 33.: Tempesti smyčka

Rreplikační smyčka tedy vykonává i jednoduchý program, který vypisuje do smyčky text. Na následujícím obrázku 34 je naznačena replikace a provedení výpisu. Obrázek 34 jsou převzaty z [14]. První část je 240. krok. Druhá část je vyobrazena ve 341. kroku. Výpis textu provádí potomek.



Obrázek 34.: Replikace Tempesti smyčky

Tato smyčka je tvořena vnitřním a vnějším pláštěm. Vnější plášť obsahuje čtyři brány, tyto brány jsou otevřené při započetí a uzavřou se po dokončení replikace. Tato smyčka se tedy replikuje do všech možných směrů zároveň. To zvyšuje výrazně efektivitu replikace. Pokud již není místo pro sebereplikaci, příslušná brána se uzavře a neprovádí se replikace v daném směru. Tento fakt řeší jeden ze základních nedostatků Langtonovy smyčky, a to funkčnost v omezeném prostoru. Tento požadavek vychází z reálných implementací a z faktu, že velikost paměti i místa na čipu je omezená.

Replikace je provedena tak, že rameno vytvoří vnitřní plášť o stejné velikosti jako originál a poté je na něj zkopírován obsah vnějšího pláště.

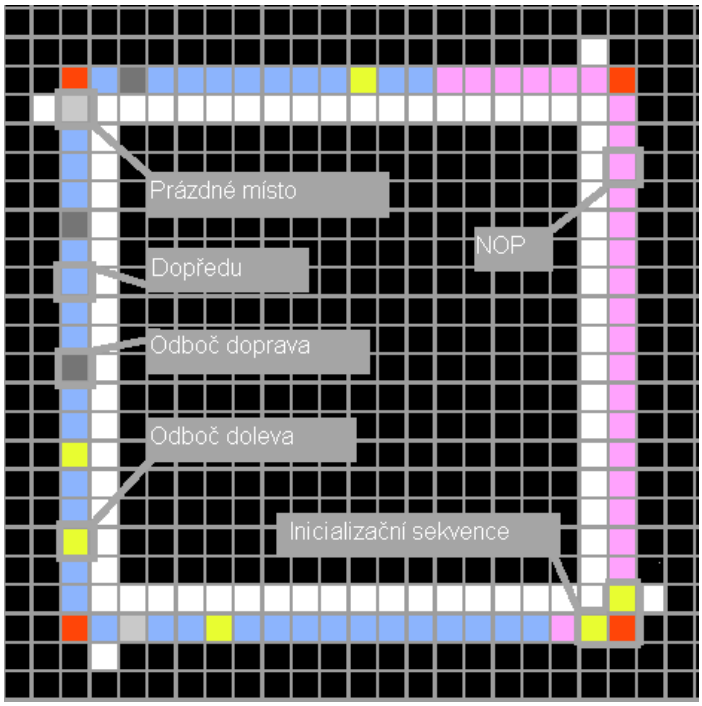
Další výhodou této smyčky je, že „nezemře“. Jakmile je dosaženo zdvojení, začne se provádět program, který vytvoří užitečný výpočet. Je to díky tomu, že údaje zůstávají i po dokončení reprodukce neporušeny. Také by bylo teoreticky možné smyčku upravit tak, aby i po uzavření bran mohly být otevřeny a smyčka by mohla opravit okolní struktury nebo by tato smyčka mohla být opravena okolními strukturami.

Vidíme, že se jedná o kombinaci Langtonovy smyčky a Von Neumannova automatu. Je zřejmé, že složitost Langtonovy smyčky není tak velká jako u Von Neumannova automatu. Složitost Tempesti smyčky není tak jednoduché určit. Je ovlivněna především cirkulujícími daty, jenž ovlivňují úkol, který má smyčka po reprodukci vykonat. Počáteční konfigurace a rozmístění prvků je tím složitější, čím složitější má smyčka vykonávat úkol. Mění se tím i rozměry této smyčky.

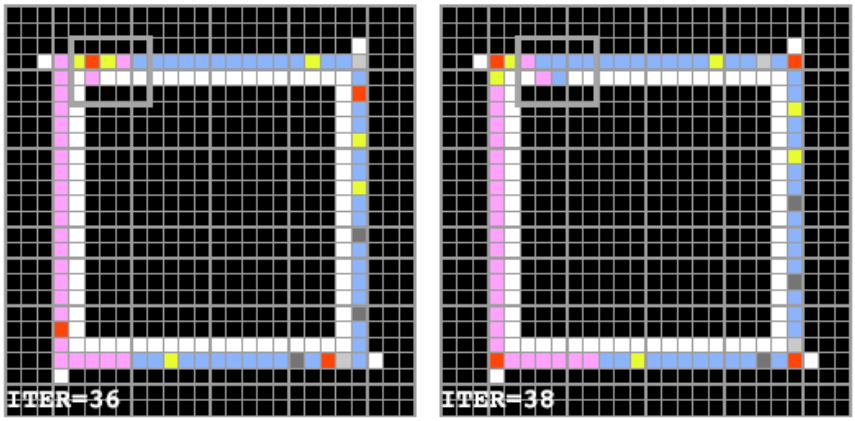
Program, jenž smyčka provádí (výpis LSL), po replikaci není na první pohled nijak užitečný, nicméně demonstruje činnost, jenž by jednou mohla být skutečným přínosem. Pokud tedy nazveme posloupnost pravidel, jež vypisují tyto znaky programem, používá tento program pět příkazů. Jsou to

instrukce vypisuj dopředu, odboč doleva, odboč doprava, prázdné místo a NOP (žádná operace). Ve smyčce je tento program zapsán na vnějším plášti a je zobrazen na obrázku 35. Samotná funkce programu je jednoduchá, inicializační sekvence dosáhne levého horního rohu a dostane se do kontaktu s takzvanými „dveřmi“ ve vnitřním plášti. Inicializační sekvence otevře tyto dveře. Tento proces je zachycen na obrázku Obrázek 36. Program se dále vykonává tak, že rotuje kolem smyčky a vstupuje přes dveře do vnitřku smyčky. Při průchodu dveřmi je zdvojen a uvnitř smyčky vykonáván. Tento proces zachycuje obrázekObrázek 37. Jakmile je dosaženo první instrukce NOP, jsou dveře uzavřeny. V této fázi je již zdvojení dokončeno a uvnitř je vypisován text. V plášti se nastaví příznak, který říká, že výpis byl již proveden.

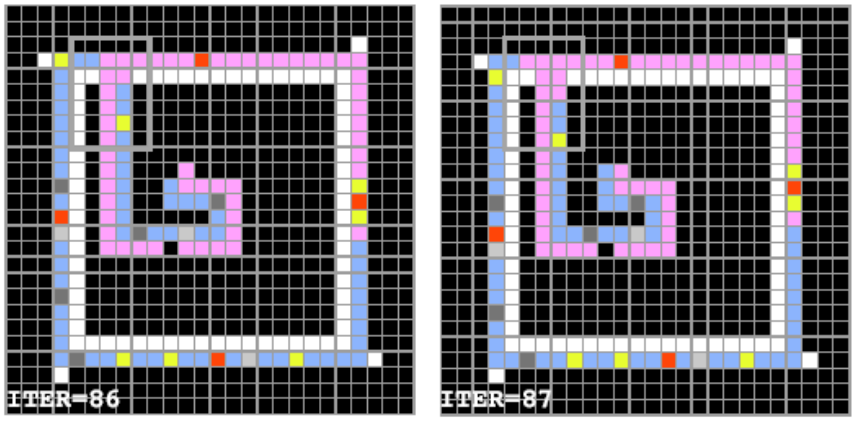
Tato smyčka se tedy snaží řešit základní nedostatky Langtonovy sebereplikační smyčky. Výsledky ukázaly, že jsou tyto smyčky schopné provádět kromě sebereplikaci i jisté další úkoly a jsou schopny pracovat v konečném prostoru. Tento text, stejně jako obrázky uvedené na následujících stránkách vychází ze zdroje [15], kde je možné najít podrobnější popis této problematiky.



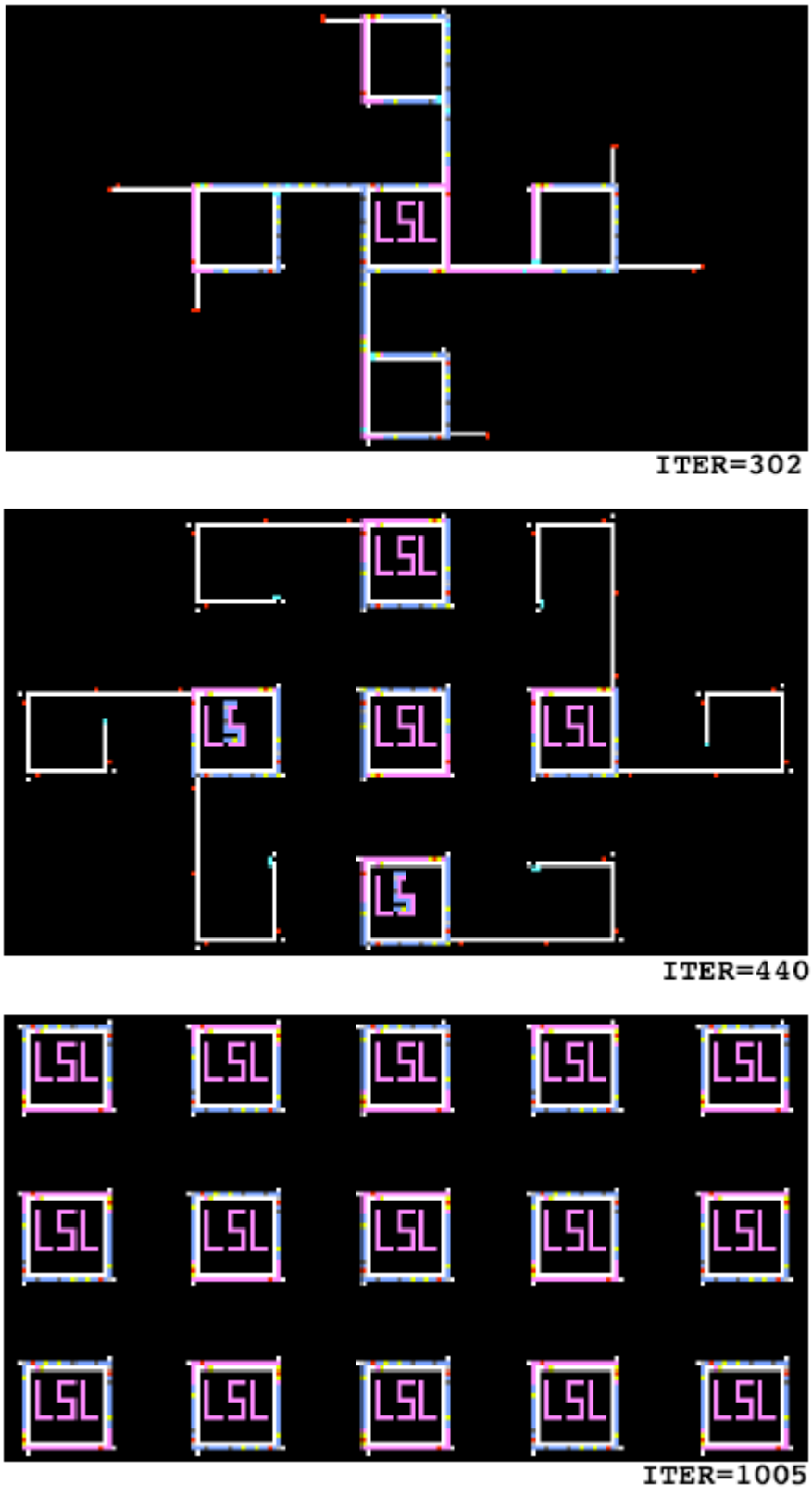
Obrázek 35.: Tempesti smyčka a popis programu



Obrázek 36.: Tempesti smyčka, posun inicializační sekvence a otevření dveří



Obrázek 37.:Tampesti smyčka, provádění programu

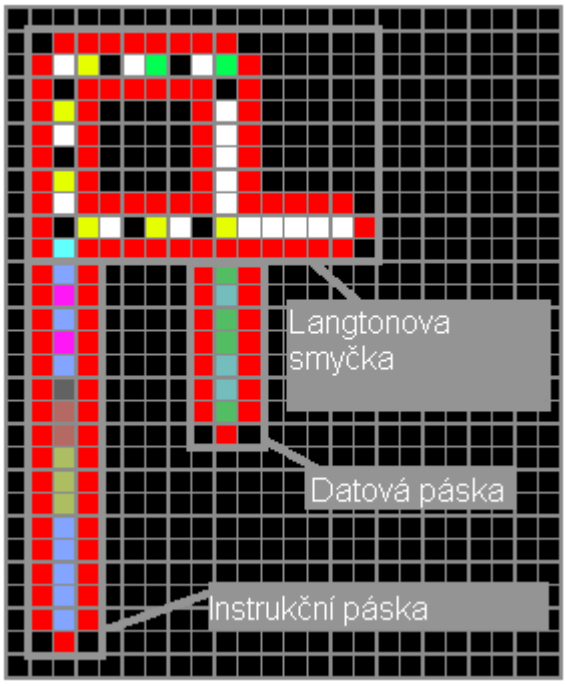


Obrázek 38 .: Tempesti smyčky, replikace potomstva

4.8 Perrierovy smyčky

Jak již bylo uvedeno v dříve (kapitola 4.7), Langtonova smyčka je ideálním příkladem sebereplikační struktury. Jednou z jejich hlavních nevýhod je ale to, že pro replikaci předpokládají nekonečný celulární automat (nekonečně velké celulární pole). Pokud tedy není dostatek prostoru pro sebereplikaci smyčky, stává se tato smyčka nefunkční. Na první pohled se může zdát tato nevýhoda poměrně triviální. Úprava smyčky, která by řešila tento problém, není až tak jednoduchá. Automat a smyčka nemá předem informace o umístění hranic (tato vlastnost vychází z definice celulárního automatu, buňka by měla mít znalosti jen o svém bezprostředním okolí). Smyčka by tedy nejprve musela ověřit, zdali je dostatek místa pro sebereplikaci nebo v případě, že narazí na hranice, zrušit již vystavěnou část ramene a vrátit se do stavu před započetím replikace. Libovolná takováto změna by vyžadovala značný zásah do struktury Langtonovy smyčky.

Perrierovy smyčky se pokouší o rozšíření Langtonovy smyčky, která pracuje s konečným prostorem a navíc se pokouší krom sebereplikace vnést do struktury i jiné funkce. Jedná se vlastně o dvoupáskový Turingův stroj přiložený k Langtonově smyčce. Jedna páska je instrukční a druhá datová. Tuto smyčku vyvinul J. Y. Perrier ve spolupráci s profesorem J. Zahndem. Část, která připomíná Langtonovu smyčku, slouží jako jakýsi dopravce Turinguva stroje. Tato kopie je však omezena pouze na jeden rozměr, protože druhý rozměr je zabrán Turingovým strojem. Při dokončení replikace je aktivován tento Turingův stroj. Následující obrázek 39 je převzat z [15] a zobrazuje Perrierovu smyčku.

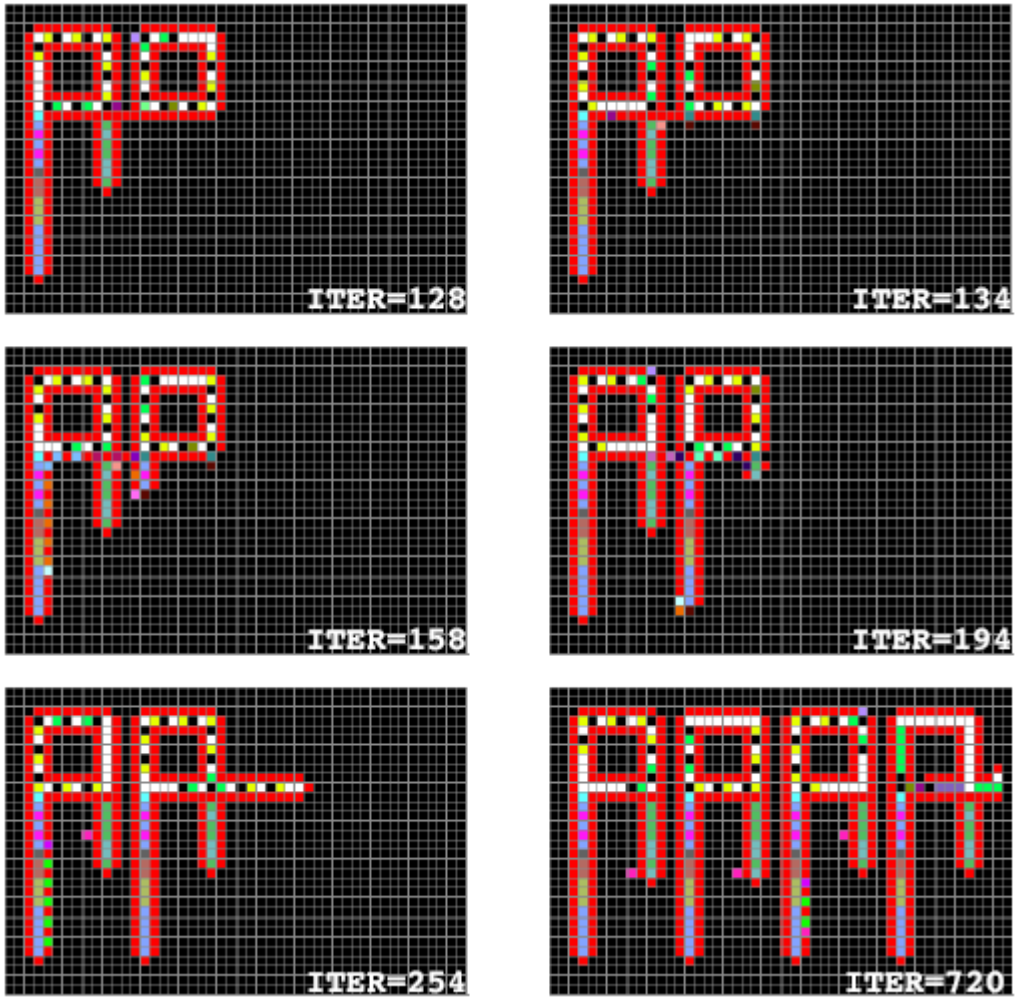


Obrázek 39.: Perrierova smyčka

Turingův stroj je sám o sobě mocný nástroj, pro implementaci v celulárním systému je však poměrně složitý. Pro jeho realizaci muselo být přidáno značné množství stavů. Stejně tak muselo být

přidáno velké množství dodatečných přechodových pravidel. Tato složitost je sice proti von Neumannovu univerzálnímu konstruktéru poměrně triviální, nicméně je značná. Z tohoto důvodu se upustilo od přizpůsobení Langtonovy smyčky pro rozšíření výše popsaných problémů.

Na sadě následujících snímků (Obrázek 40) je zachycena sebereplikace Perrierovy smyčky. Smyčka se replikuje tak dlouho, dokud nenarazí na konec celulárního prostoru. Obrázky jsou převzaty ze zdroje [15], zde je také možné nalézt podrobnější popis této problematiky.

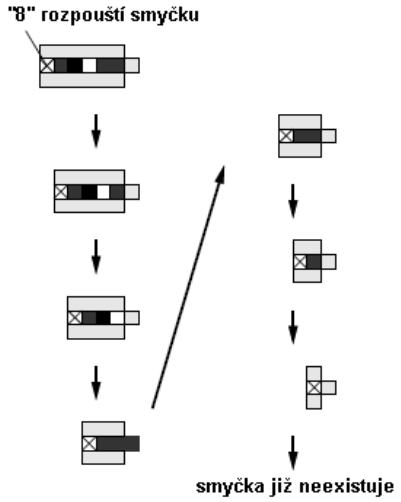


Obrázek 40.: Sebereplikace Perrierovy smyčky

4.9 Seberperodukující se smyčky s rozpustnou strukturou (SDSR-Loop)

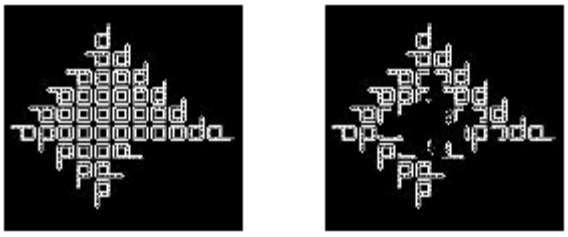
Structurally dissolvable self reproducing loops (SDSR-Loop) jsou rozšířením Langtonových smyček, které provedl Hiroky Sayama [2]. Rozšíření spočívá v přidání stavu označeného „8“ (rozpouštěcí stav). Pokud již smyčka není schopna replikace a vzniká „mrtvá struktura“ nebo dojde k chybě při

replikaci, dochází k jejímu rozpadu a tím k uvolnění místa pro žijící smyčky. Následující obrázek 41 je převzat z [2] a demonstruje funkci nového stavu.



Obrázek 41.: Rozpouštějící se smyčka

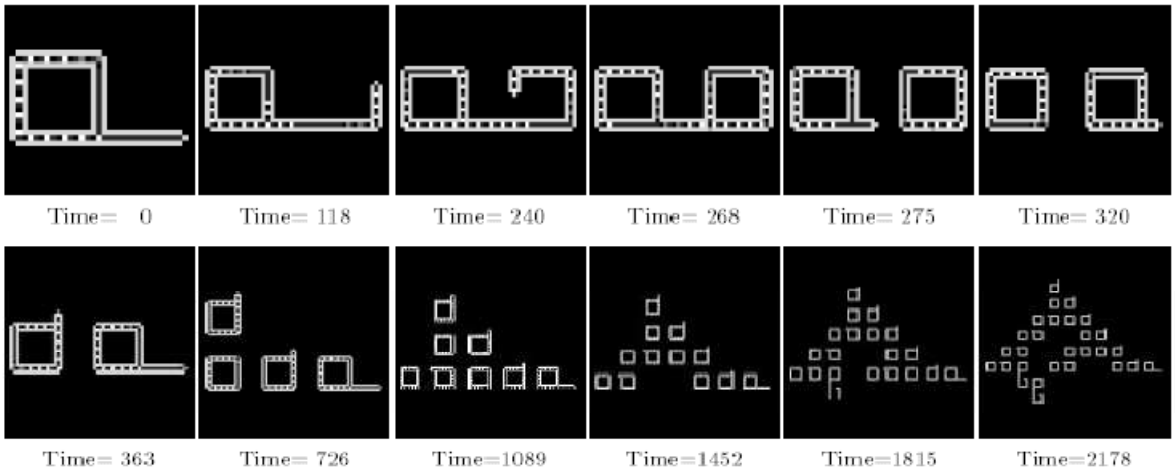
Při použití nově zavedeného stavu a rozšíření pravidel je možné sledovat mnohem zajímavější chování, které je pro skutečný život mnohem typičtější, než je možné pozorovat u Langtonových smyček. Emergentní chování takového systému je zachyceno na obrázku 42. V levé části je zobrazena kolonie SR-smyček a v pravé SDSR-smyček. Je zřetelně vidět, že mrtvé smyčky v pravém obrázku 42 zanikají a mizí. Obrázek 42 je převzat ze zdroje [2].



Obrázek 42.: SR-Loops vs. SDSR-Loops

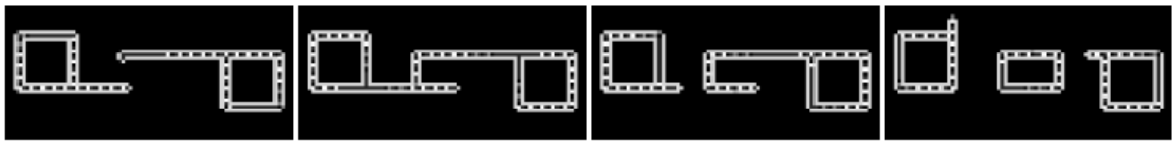
4.10 Evoluční smyčky (Evoloop)

Jedná se o úpravu SDSR-smyček. V rámci disertační práce ji vytvořil Hiroky Sayama [2]. Jedná se vlastně o evoluční systém SDSR-smyček. Tyto smyčky jsou upraveny tak, že jejich chod spíše připomíná Coddův automat (4.3). Velikost těchto smyček se přizpůsobuje velikosti volného prostoru (rameno vytvářející smyčku se může začít stáčet v závislosti na volném prostoru). V případě, že není možná replikace, smyčka se rozpadne. Pokud dojde ke srážce smyček, smyčka stáhne své rameno nebo se rozpadá. Tyto smyčky mají 9 stavů a pracují s von Neumannovým okolím. Replikace této smyčky je zachycena na následujícím obrázku 43, ten je převzat z [2].



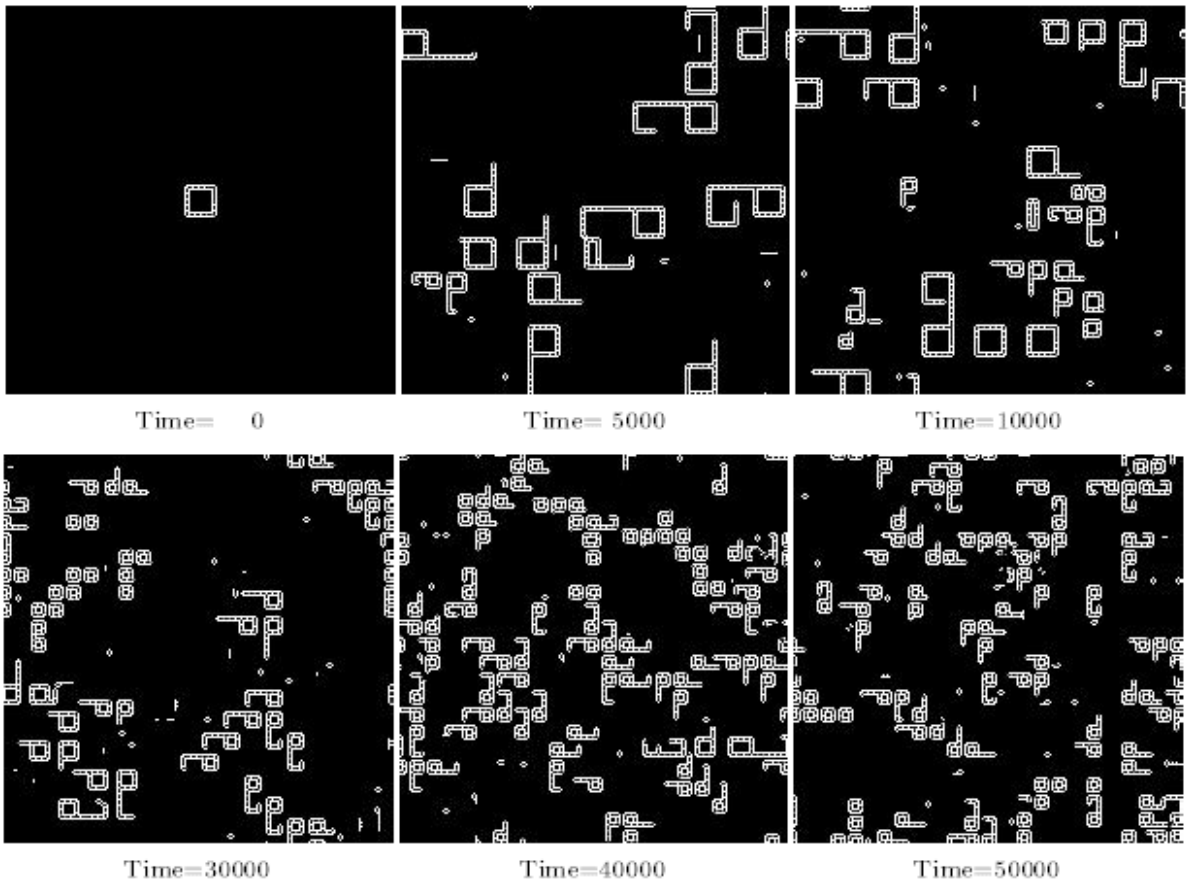
Obrázek 43.: Evoloop

Následující obrázek 44 zobrazuje kolizi dvou ramen smyčky. Rameno pravé smyčky se srazí s ramenem levé smyčky. Levá smyčka detekuje kolizi a rameno se stáhne zpět do smyčky. Nestáhne se však celé. Část ramena levé smyčky se stane součástí ramena pravé smyčky. Je vytvořena nová smyčka. Tento jev je mimořádně zajímavý. Dojde zde ke křížení dvou smyček a předání genetického materiálu z jedné smyčky do druhé. Jde vlastně o primitivní „sexuální kontakt". Tento fakt se stal základem pro takzvané sexyloop, které jsou popsány v další části. Následující obrázek 44 je převzat z [2].



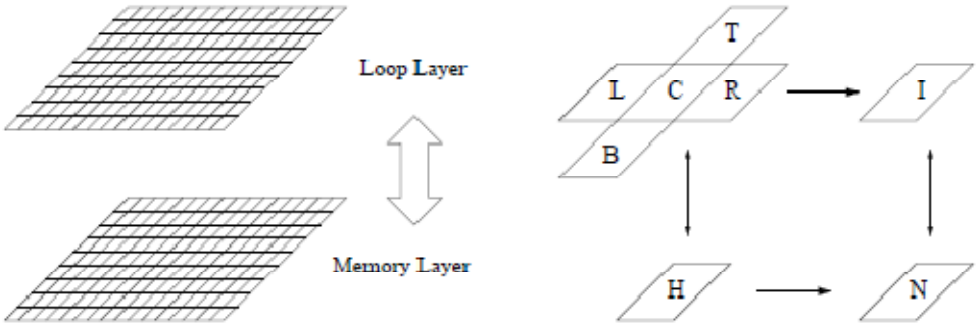
Obrázek 44.: Evoloop a kontakt dvou smyček

Při simulaci se zjistilo, že se pravidelně vyskytují smyčky různých velikostí. Podle těchto velikostí můžeme smyčky rozdělit do několika skupin [2]. Simulace také ukázala, že se nejlépe daří malým smyčkám, které se v průběhu simulace nejlépe množí. Na následujícím obrázku 45 je zachycen průběh simulace, která potvrzuje předešlé tvrzení. Obrázek 45 je převzat z [2].



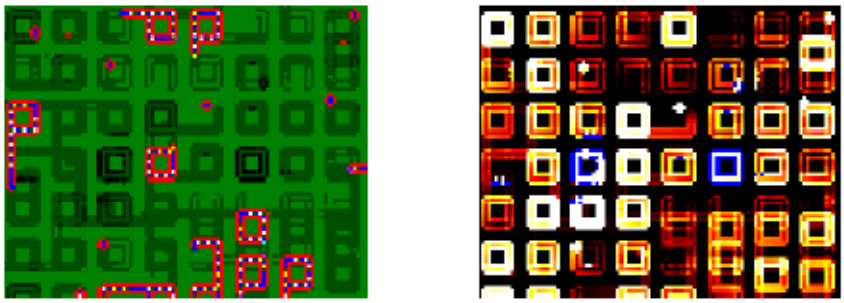
Obrázek 45.: Vývoj simulace evoloop

U těchto druhů smyček je možné pozorovat formování populací. Ze všech dříve uvedených struktur má tato nejblíže ke skutečným živým organismům a stává se tedy jednou z nejvhodnějších struktur pro simulaci života. Proto, abychom mohli zkoumat vývoj těchto struktur z dlouhodobého hlediska (dlouhodobá simulace), je potřeba zavést určitá rozšíření. Již jsem se zmínil, že v simulaci evoloop se dařilo nejlépe malým jedincům (Hiroky Sayama je označil jako druh „4“), u kterých replikace trvala krátkou dobu. Pro větší rozmanitost ve sledování vývoje populace vytvořila skupinka z univerzity v Amstrdamu nový experiment. Autory jsou Daniela Gavidia, Chris Salzberg a Antoni Antony. Do světa evoloop přidaly pojmy potravy a choroby. Sledování těchto dvou elementů vyžadovalo přidání nové mřížky. Tato mřížka měla stejné rozměry jako mřížka automatu pro evoloop. Jednotlivé buňky byly umístěny tak, aby si navzájem korespondovaly. Pravidla pro evoloop využívala von Neumannovo okolí. Nová mřížka s okolím nepracovala. Tato mřížka byla nazvána paměťová vrstva (Memory Layer). Idea je znázorněna na následujícím obrázku 46, ten je převzat z [18].



Obrázek 46.: Rozhraní mezi vrstvou smyček a paměťovou vrstvou

Tuto myšlenku měl i Sayama, ten ale uvažoval, že buňka paměťové vrstvy bude mít pouze dva stavy (jeden bit). Tato myšlenka však byla rozšířena. Buňky paměťové vrstvy byly implementovány jako čítač. Tento čítač pak slouží k udržení informace o čase výskytu života na dané buňce a určuje tak množství potravy nebo výskytu chorob na příslušné buňce mřížky evoloop. Pravidla této skryté vrstvy jsou velmi jednoduchá a je možné je najít v [18]. Princip se pokusím přiblížit na režimu zaznamenávajícím množství potravy. Ilustrace je na následujícím obrázku 47, ten je převzat z [18].



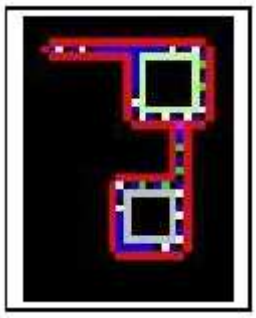
Obrázek 47.: Evoloop a paměťová vrstva

V pravé části obrázku 47 je zobrazena paměťová vrstva, která zobrazuje množství potravy při provádění simulace. Černá barva značí maximum potravy. Světlé odstíny značí ubývající množství a modrá barva, že veškerá potrava je pryč. Choroby fungují na podobném principu. Uplatňují se při rozpouštění smyček. Díky těmto vlastnostem je možné lépe studovat vývoj populací těchto smyček. Následující obrázek 48 zachycuje šíření chorob v populační explozi těchto evolučních smyček, je převzat z [18]. V tomto zdroji je také možné získat bližší informace k této problematice. Šedá barva na obrázku zachycuje výskyt chorob.

Obrázek 48.: Šíření chorob v populaci (705K. 710K, 715K ITER)

4.11 Sexyloop a F-Sexyloop

Autorem těchto smyček je Nikolas Oros, který v roce 2006 upravil Saymovy evoluční smyčky (evoloop). Práci vedl profesor Chrystopher L. Nehaniv. Tyto smyčky pracují s von Neumannovým okolím. Smyčky, jak již název napovídá, mají schopnost „sexu“. Díky tomu jsou smyčky schopné přenosu svého genetického materiálu do jiných smyček.



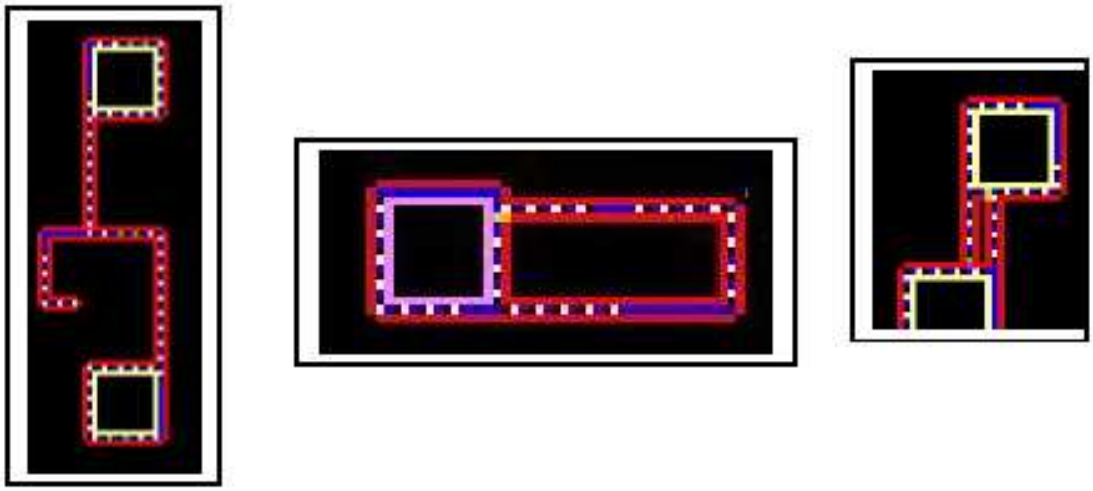
Obrázek 49.: Sexuální spojení, spodní smyčka "napadla" vrchní smyčku a dojde k přenosu genetického materiálu

K evolučním smyčkám byl přidán nový stav a nová pravidla. Práce ukázala, že v evolučních smyčkách vzniká evoluční proces jen kvůli tomu, aby se smyčky přizpůsobovaly svému okolí. Nikolas Oros vytvořil dva různé modely sebereplikujících se smyček s pohlavím, sexyloop M1 a M2. Důvodem bylo posouzení vlivu pohlaví na evoluční proces prostřednictvím srovnání evoloop a sexyloop. Výsledkem bylo zjištění, že sexy loop M1 a M2 vytvořily mnohem rozmanitější a rychleji se vyvíjející populaci než v případě evoloop procesů s bezpohlavními protějšky. Nejzajímavější situace nastala v modelu M2, kde díky evoluční dynamice vznikly velmi rozdílné typy chování. Nejzajímavějším fenoménem bylo však to, že evoluční proces vybral poměrně rychle k reprodukci jedince většího typu. U těchto jedinců reprodukce trvá mnohem více času než u malých jedinců, které upřednostňovaly modely M1 a evoloop. Z těchto poznatků vyplývá, že v modelu M2 nedominovali jedinci, kteří se množili nejrychleji, ale ti, jenž se nejlépe a nejrychleji přizpůsobili pohlavnímu rozmnožování. Tyto výsledky daly první důležité poznatky o sebereplikujícím se systému v celulárních automatech, kde sex hraje významnou roli [11].

V roce 2008, během druhého roku své práce na disertaci, rozšířil Nikolas Oros sexyloop na

F-sexyloop. Byly přidány dva stavy. Tyto smyčky nesou gen usnadňující přenos genetického materiálu. Tento gen je analogický k faktoru F plasmidu bakteriální konjunkce [12]. Tento faktor přispívá schopnosti působit jako dárce genetického materiálu. V celé řadě případů tento gen přetrvává v genomu dominantních jedinců během evolučního procesu [12].

Na následujícím obrázku 50 jsou znázorněny „sexuální kontakty" smyček. V levé části je znázorněn přesun genetického materiálu z ramene jedné smyčky do ramene jiné smyčky. Ve středu je zobrazena smyčka, která přesouvá genetický materiál sama do sebe. Na obrázku 50 v levé části jsou zachyceny smyčky, které přesouvají genetický materiál navzájem jedna z druhé. Obrázek 50 je převzat ze zdroje [12].



Obrázek 50.: Sexyloop a přenos genetického materiálu

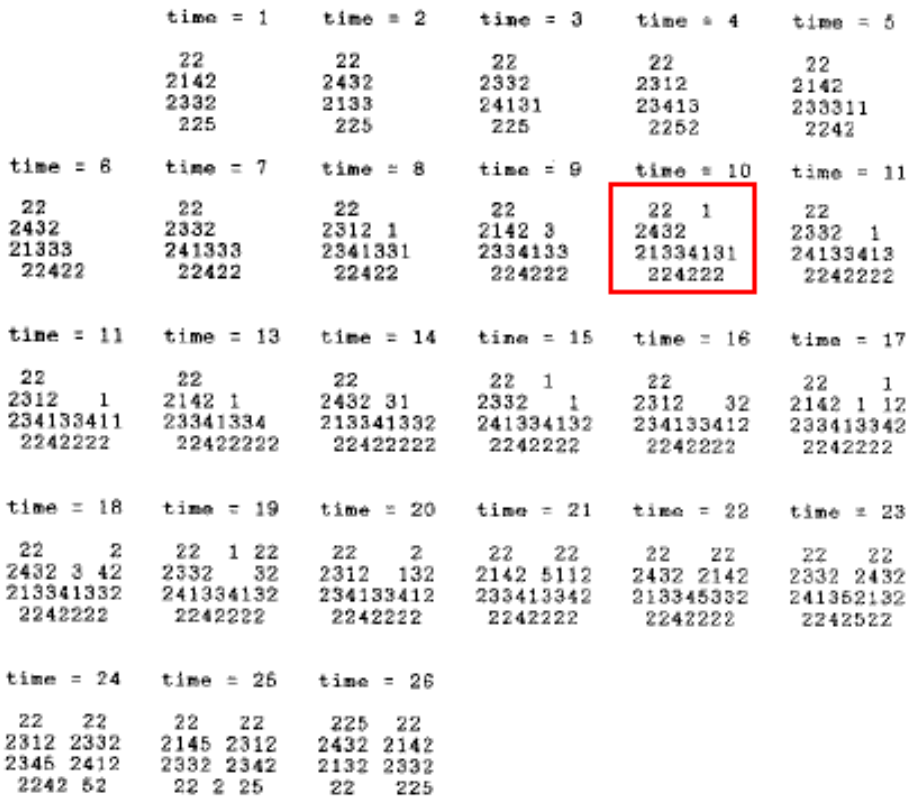
5 Urychlení sebereplikace

V této části se budu zabývat urychlením sebereplikace v celulárních systémech. V této kapitole se pokusím odpovědět na následující otázky. Proč vlastně sebereplikaci urychlovat? Je vůbec možné sebereplikaci urychlit, a pokud ano, za jakou cenu? Na první z otázek se pokusím opovědět již v tomto textu. Jak již bylo naznačeno dříve, sebereplikace je důležitou součástí každodenního života. Je všude kolem nás, některé buňky se množí příčným dělením, čímž vlastně vytvoří kopii sama sebe. V současné době se do popředí dostává klonování, což je vlastně sebereplikace a pokud k tomuto problému přistoupíme obecněji, můžeme říct, že i narození lidského potomka je jistou formou replikace dvou jedinců. Moderním trendem je urychlování všeho, co přináší užitečný výsledek až ve formě zralosti. Z tohoto pohledu se může zdát, že by mohlo být užitečné urychlit i sebereplikaci. Tato myšlenka je sice velmi lákavá, ale nese sebou i mnohá úskalí. V mé práci se nebudu zaměřovat na urychlení rození dětí, množení buněk nebo klonování (i když možná i k tomuto dá jednou tato práce, nebo práce podobné, pozitivní podnět). Zaměřím se na urychlení sebereplikujících se struktur v celulárních automatech. Urychlení replikace těchto poměrně jednoduchých a na první pohled neužitečných struktur samo o sobě žádný užitečný výsledek přímo nepřinese, ale mohlo by se stát vzorem pro mnohem užitečnější mechanismy, které by využívaly tohoto principu. Své místo by mohly najít například v simulaci umělého života, v náročných výpočtech, nebo i v kosmických programech. V roce 1980 v NASA začal tým s názvem Self Replicating System Concept Team zkoumat možnosti využití sebereplikace v kosmickém programu. Hlavní myšlenkou byla již dříve zmíněná kolonizace jiných světů [19]. Jejich vize byly ohromné, téměř hraničily se science-fiction. Od obrovských lunárních továren (Obrázek 51) šířících se po povrchu měsíce až po reálnější reprodukční sondování napříč galaxií. Technologie, která by tyto plány umožnila, je sice ještě velmi vzdálená, ale už tehdy se rozvíjely teoretické problémy, které by přispěly k rozvoji těchto systémů. Tým navrhl lunární továrnu, která by v jisté formě prováděla svoji replikaci podobně jako struktury v celulárních automatech. Tak by byl kolonizován a osídlen měsíc. Poznamenejme, že Langton svoji strukturu objevil až v roce 1984 a tým NASA byl rozpuštěn již v roce 1983 kvůli nedostatečným finančním prostředkům. Pokud by se tedy podařila urychlit sebereplikace v celulárních systémech, mohlo by to například vést k urychlení kolonizace měsíce.

Obrázek 51.: Návrh sebereplikující se lunární továrny

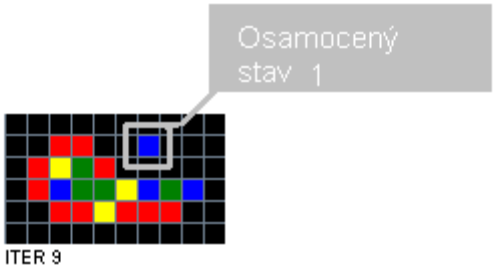
5.1 Urychlení sebereplikace v celulárních systémech

V následujícím textu práce se budu zaměřovat na urychlení sebereplikace v celulárních systémech a to především urychlení replikace Bylovy smyčky. V předcházejícím textu jsem se pokusil obhájit myšlenku užitečnosti urychlení sebereplikace. Nyní vyvstává následující otázka. Jak vlastně sebereplikaci urychlit? Způsobů urychlení je celá řada. Přirozeně se dá do urychlování zařadit i zvyšování výpočetního výkonu a tím i urychlování simulací. Toto urychlení však nebudeme v této práci dál uvažovat, protože nezmenšuje počet kroků, které jsou v celulárním simulátoru potřeba k provedení sebereplikace, pouze tyto kroky urychluje. Moji snahou je zmenšit počet kroků celulárního automatu. Tohoto se dá dosáhnout několika způsoby. Jedním z nich by bylo přidání nových stavů do struktury a reorganizace pravidel. Já se však zaměřím na celulární automat, který umí za daných podmínek změnit pravidla sebereplikace. Pro demonstraci tohoto řešení jsem si vybral takzvanou Bylovu smyčku, kterou jsem popsal v předcházející části této práce (4.5). Smyčka pracuje se 6 stavy a replikace je hotová v 25 krocích. Tuto smyčku jsem volil záměrně, díky její relativně malé složitosti (s porovnáním oproti Langtonově smyčce) a také díky ideálnímu stylu sebereplikace. Replikace je kompletně zachycena na sadě následujících obrázků, ty jsou převzaty z [2].



Obrázek 52.: Kompletní sebereplikace Bylovy smyčky

Sebereplikaci této smyčky jsem prostudoval a pro změnu pravidel jsem zvolil zvýrazněný stav na obrázku 52. Jedná se o 9. krok (někdy označován jako 10., protože různé zdroje vychází z různých počátečních inicializací této smyčky) replikace. Tento krok je velmi výhodný díky stavu replikace smyčky. Pro přehlednost je stav simulace v tomto kroku zachycen na následujícím obrázku 53. Struktura má v této fázi velmi vhodný tvar pro přestavbu pravidel. Je to hlavně díky osamocenému stavu „1“, který je na obrázku zvýrazněn. Také je zde ideálně natočeno jádro, které by při pokračování v replikaci klasickým způsobem dál rotovalo.



Obrázek 53.: Klíčová fáze pro změnu pravidel

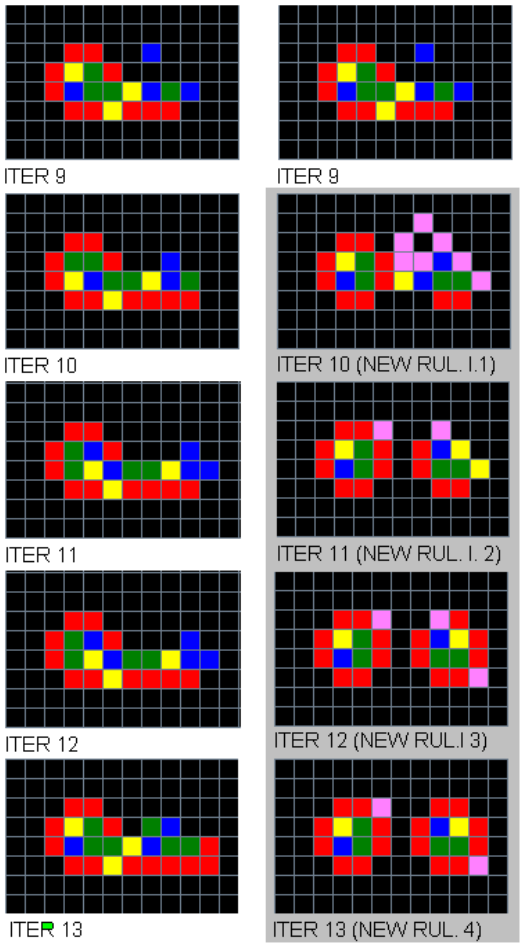
Z tohoto stavu jsem vycházel a začal jsem sestavovat pravidla, která by urychlila tuto replikaci. Sestavení těchto pravidel není triviální záležitostí, protože nová pravidla musí dotvořit struktury do výsledného tvaru. Pravidla musí fungovat pro různé natočení smyčky a nesmí ovlivňovat původní strukturu smyčky. Také jsem se snažil, aby se co nejméně lišila od pravidel původních. Tuto kombinaci se podařilo objevit. Nová pravidla dotvoří smyčku ve čtyřech krocích. Průběh je znázorněn na obrázku Obrázek 54. V levé části je smyčka, která se replikuje podle původních pravidel, a v pravé je smyčka, která v desátém kroku již využije nová pravidla. Replikace je tak urychlena z 25 kroků na 13, což je 52% původní doby. Tedy urychlení o 48%.

V tomto okamžiku již mohu odpovědět na první část druhé otázky, kterou jsem položil v úvodním textu této kapitoly. Ano, sebereplikaci je možné urychlit. A to velmi výrazně urychlit. Na druhou část druhé otázky se pokusím odpovědět v dalším textu.

Pravidla pro urychlení jsem generoval ručně, pomocí logiky a mnoha pokusů. Výrazně k této činnosti dopomohl vytvořený nástroj pro simulaci, který je součástí této práce, a který obsahuje nástroje pro efektivní generování pravidel celulárního automatu (kapitola 6).

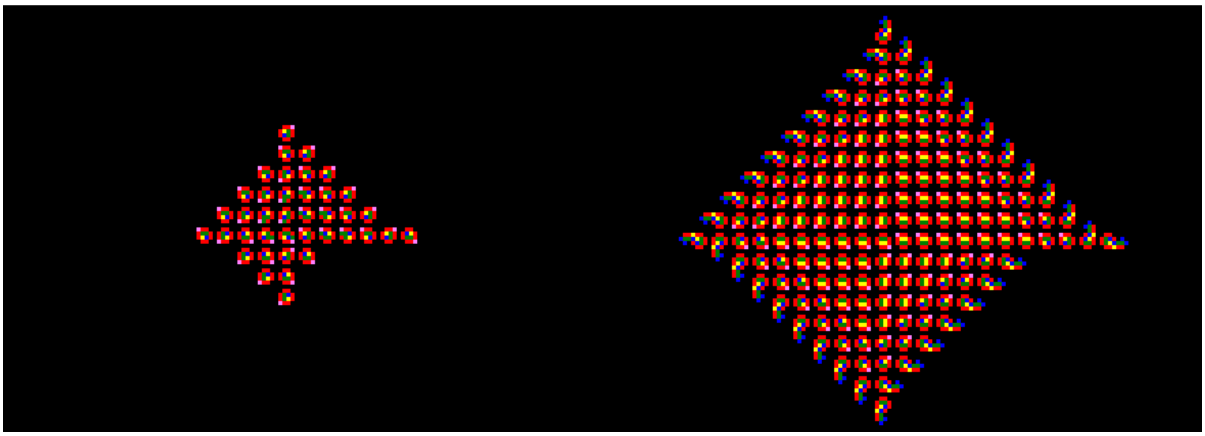
Pravidla byla vytvořena tak, aby co nejvíce korespondovala s pravidly původními. Důvod k tomuto počínání je v myšlence, že ke změně pravidel dojde za určitých okolností. Tato změna by měla být co nejrychlejší a nemělo by se jednat o změnu celé sady pravidel, ale pouze o modifikaci několika pravidel z původní tabulky. Pro přehlednost však budu v následujícím textu uvažovat dvě sady pravidel. Při vzájemném porovnání jsem bral v potaz 317 pravidel původní replikační tabulky. Jsou v nich zahrnuty všechny rotace (rotace okolí pro funkci ve všech směrech) a jsou zde i některá pravidla, která nemění stav buňky. Jsou zde zahrnuta proto, že při přechodu na novou sadu pravidel je pravá strana tohoto pravidla modifikována a již mění stav buňky. Při přechodu na novou sadu

pravidel dojde ke změně 127 pravidel z původní sady. Ostatní zůstávají stejná. Jedná se tedy o 40,6% změnu. Pravidla a vyznačené změny jsou přiloženy jako příloha k této práci.



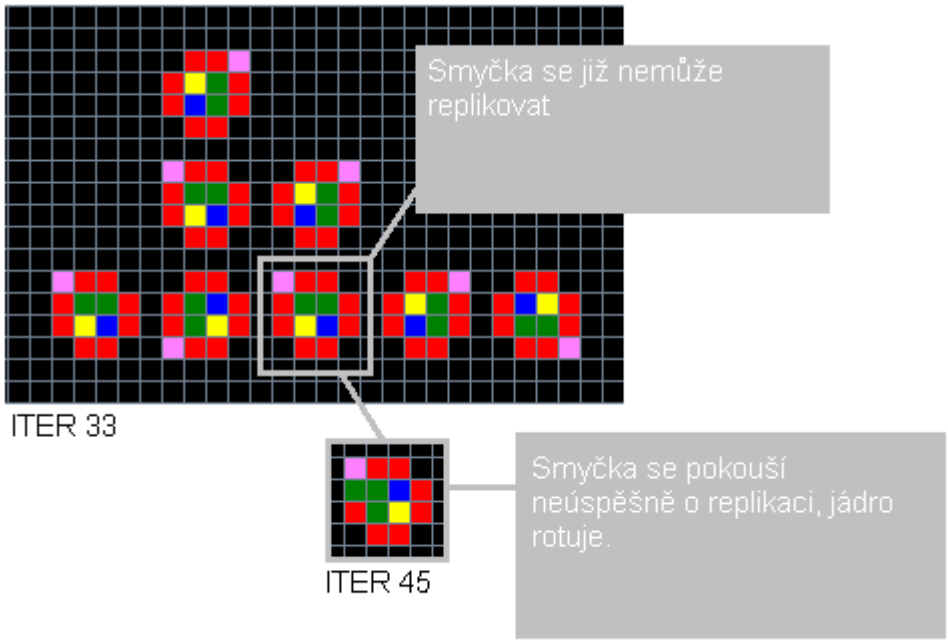
Obrázek 54.: Původní vs. urychlená replikace

Po dokončení replikace je nutné změnit pravidla zase do původního stavu a pokračovat v simulaci po dobu devíti kroků s původní sadou. Poté je možno na čtyři kroky opět změnit pravidla pro urychlení sebereplikace. Následující obrázek 55 ukazuje rozdíl neurychleného (vlevo) a urychleného (v pravo) procesu ve 150. kroku simulace.



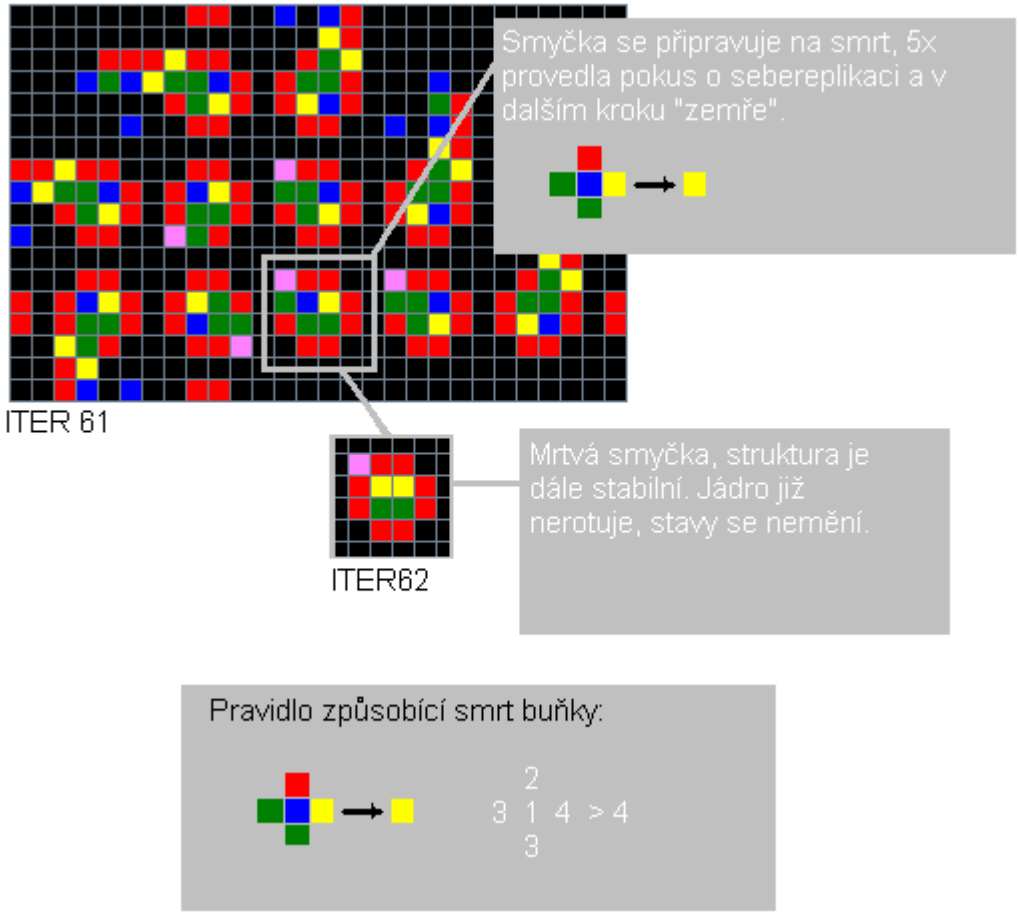
Obrázek 55.: Neurychlená vs. urychlená populace

Toto urychlení však přináší i problémy. První nevýhodou je složitější simulátor, který musí ve správnou dobu pozměnit příslušná pravidla. To je spíše implementační záležitost, a proto se jí nebudu dále zabývat. Dalším problémem je pozastavení rotace jádra během čtyř kroků, kdy je simulace řízena novými pravidly. Jádro tedy rotuje pouze v devíti krocích, kdy je simulace řízena podle původních pravidel. V devátém kroku replikace se jádro natočí do výsledné polohy (v jaké by se nacházelo v 25. kroku původní simulace) a v té setrvá do nové periody. Tím nejzávažnějším negativním prvkem urychlení je však velmi zajímavý efekt. Smyčky začínají „umírat“. Původní Bylovy smyčky jsou nesmrtelné. Jejich jádro neustále rotuje a smyčka neustále kontroluje, zdali se neuvolnilo místo pro sebereplikaci. U urychlené replikace smyček je tomu jinak. Pokud smyčka již nemá prostor pro započetí replikace, pokusí se pětkrát započít novou replikaci (pětkrát otevře bránu pro replikaci). Její jádro paralelně k pěti pokusům o replikaci provede čtyři a půl rotací a poté smyčka zemře. Pokud je simulátor inicializován na jednu smyčku, tak první smrt nastane v 62. kroku simulace. Postupně zemřou všechny smyčky, které se nemohou replikovat.

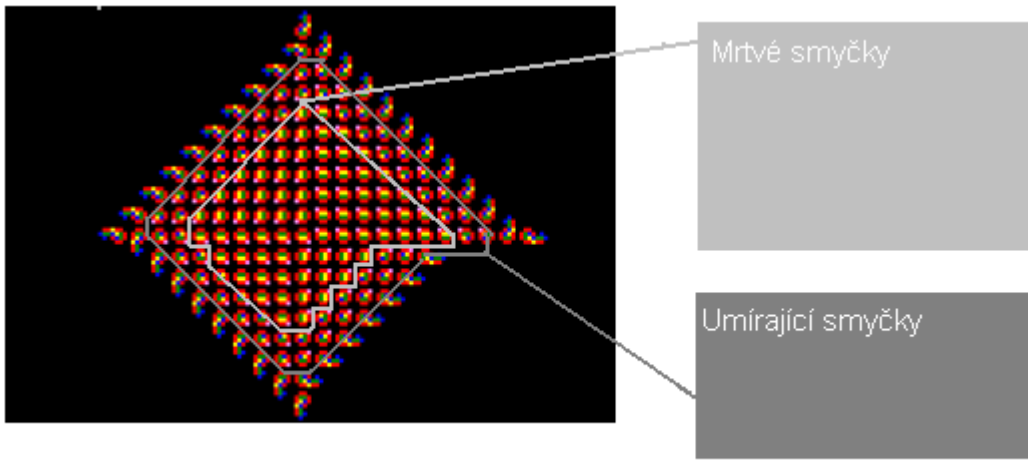


Obrázek 56.: Umírající smyčka

Smrt smyčky je vlastně ustálení její struktury. Smyčka již nemění své stavy, jádro již nerotuje a pokusy o replikaci ustanou. K tomuto jevu nastává díky výskytu speciální konfigurace stavů ve von Neumannově okolí a následné změny pravidel (na pravidla pro urychlení replikace), kde tato kombinace změní stav buňky ze stavu „1“ na stav „4“. To způsobí, že po návratu k původním pravidlům již buňka nenalezne pravidla, která by oživila smyčku. Řešení této situace by mohlo být například pomocí zavedení nového stavu a modifikací nových pravidel. Osobně považuji toto chování za velmi zajímavé.



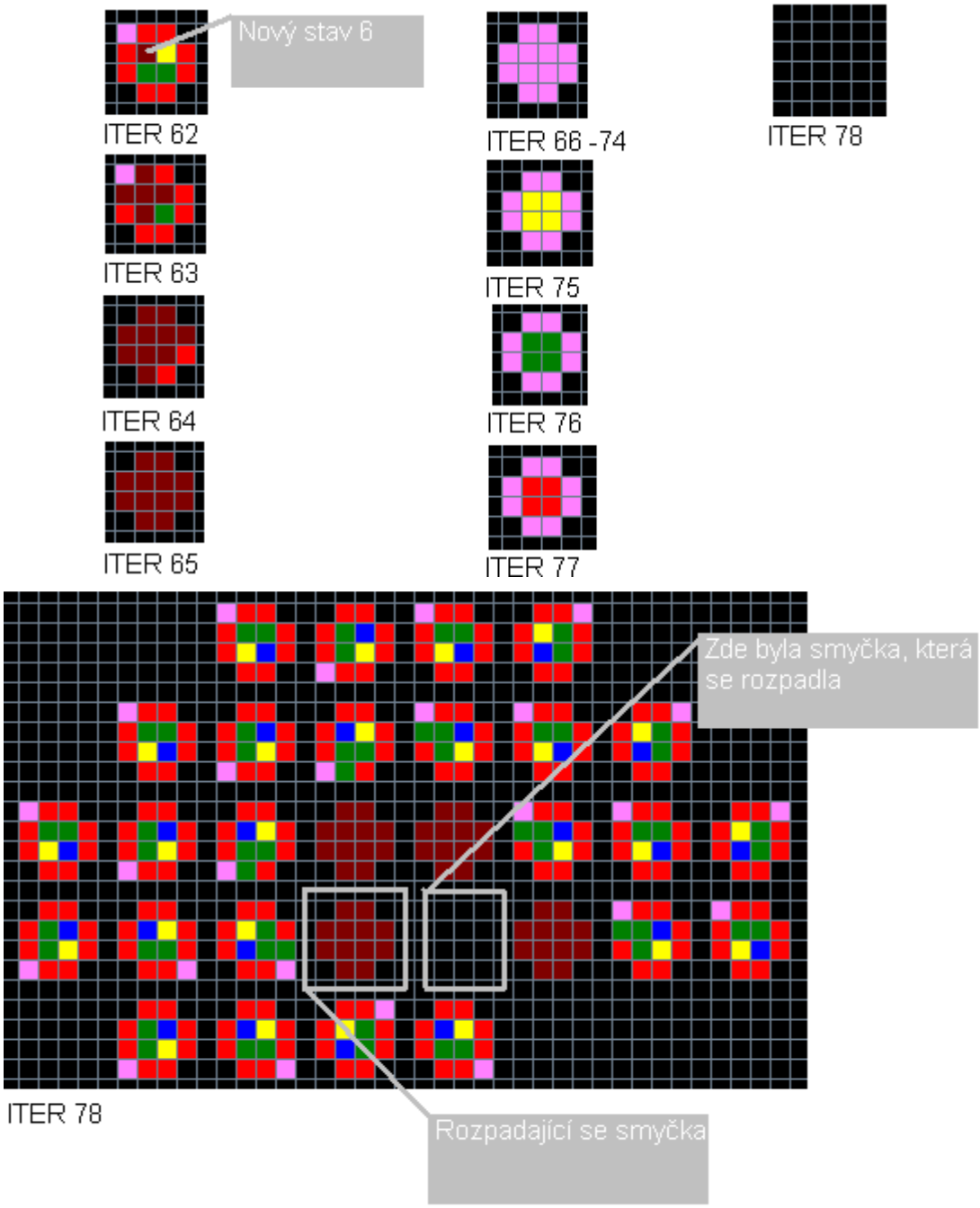
Obrázek 57.: Smrt buňky



Obrázek 58.: Populace smyček

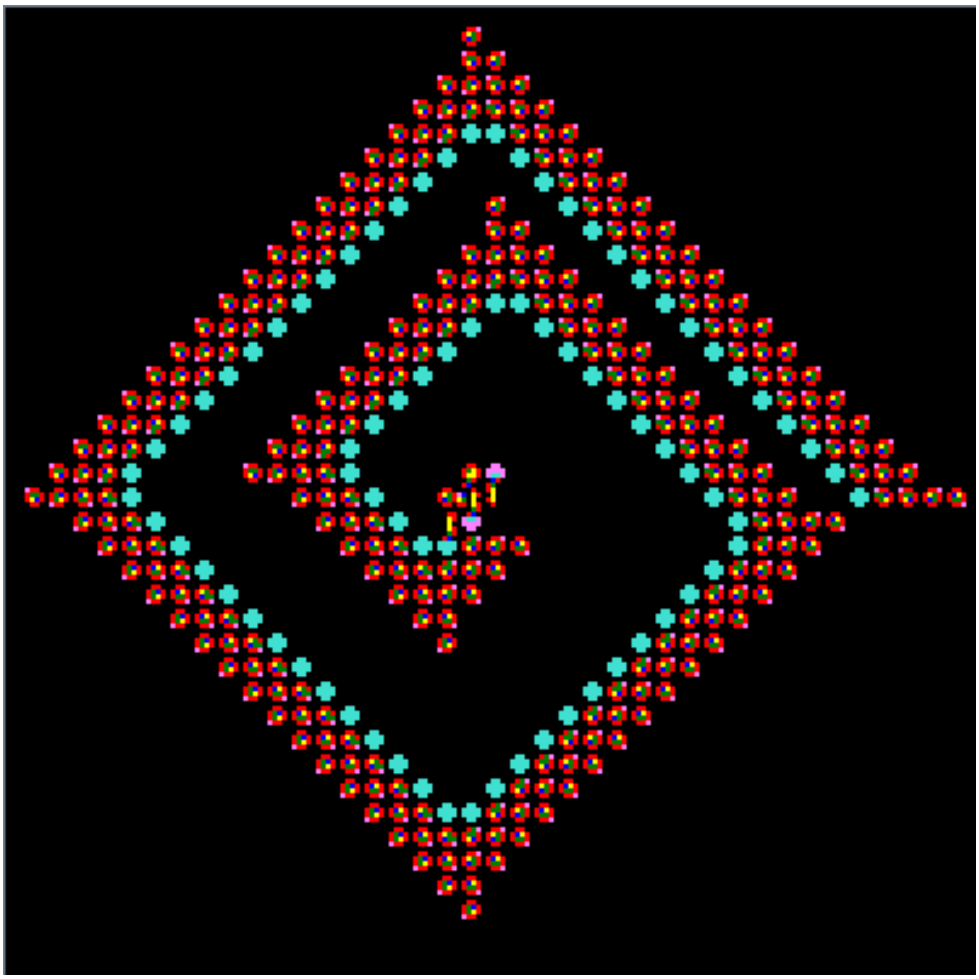
Při studii chování těchto buněk jsem se inspiroval SDSR-Loop (kapitola 4.9). Když již smyčka zemře, mohla by uvolnit svůj prostor živým smyčkám. Rozšířil jsem tedy smyčky o jeden stav. Tento stav symbolizuje smrt a následný rozpad buňky (přechod do stavu „0“). Modifikoval jsem pravidla a přidal nová, která zajistí rozpad smyčky. Základem je modifikace pravidla, které způsobuje smrt smyčky. Pravá strana pravidla se ze stavu „4“ změní na nový stav „6“. Následující obrázek 59

znázorňuje rozpad smyčky. Rozpad trvá 16 kroků. Tato doba je poměrně dlouhá. Je to proto, aby smyčka neuvolnila prostor příliš brzy a nevznikaly tak časté kolize dvou smyček, které se replikují proti sobě a jejich nedokončené repliky se srazí.

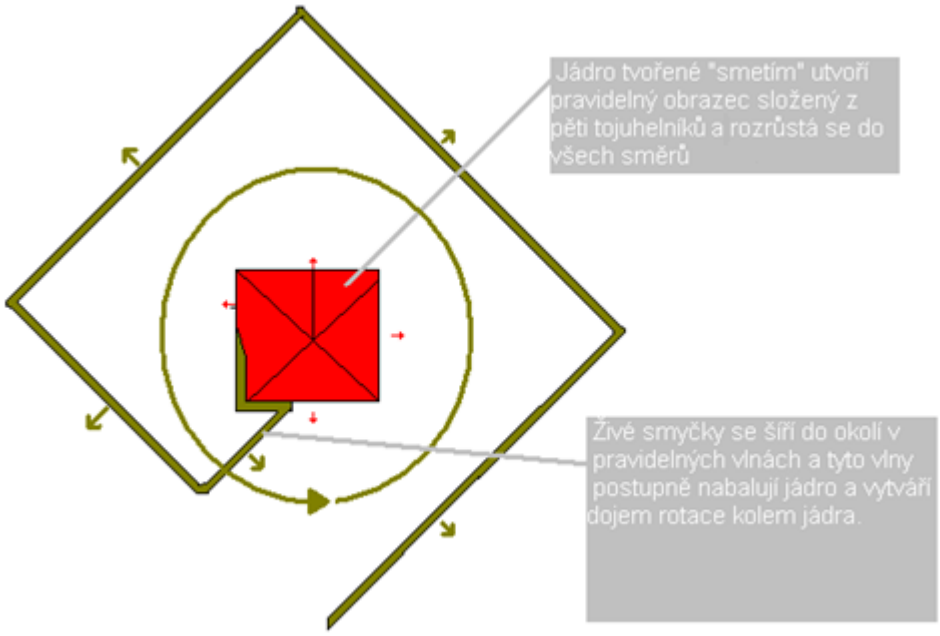


Obrázek 59.: Rozpad mrtvé smyčky

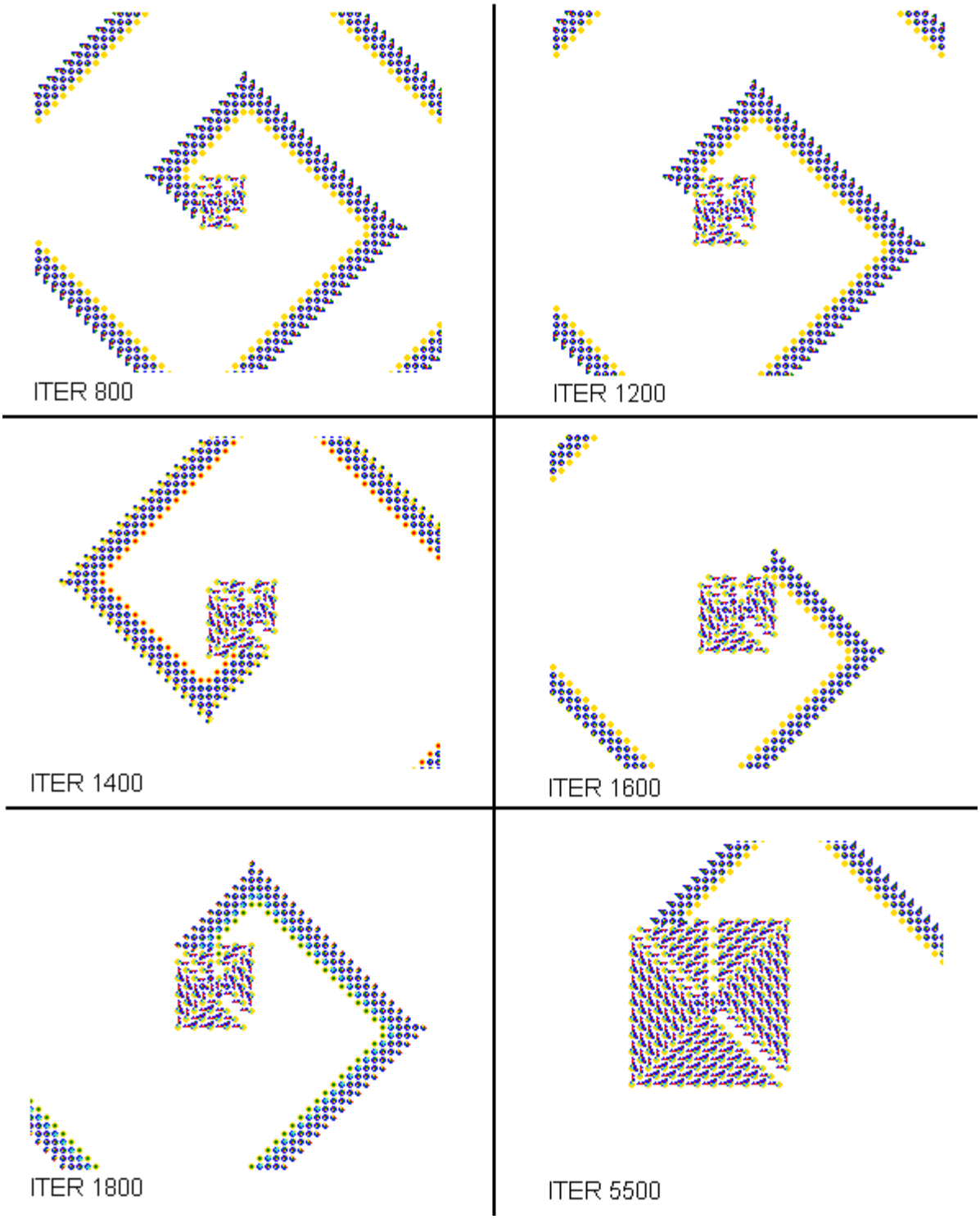
Na obrázku 59 je znázorněná populace těchto smyček v 260. kroku simulace. Je možné pozorovat, že smyčky se šíří v několika vlnách a vytvářejí pravidelný obrazec. Ve středu je možno pozorovat kolizi několika smyček, které nemohly dokončit sebereplikaci a tudíž se nemohly ani rozpadnout. Dá se říct, že zemřeli předčasně a jejich pozůstatky vytváří v jádru „smetí“.



Obrázek 60.: Populace rozpouštějících se smyček v 260. kroku



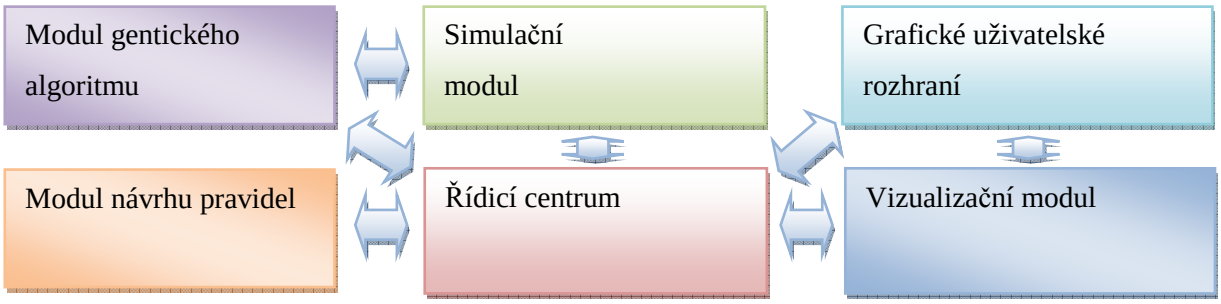
Obrázek 61.: Model vývoje simulace



Obrázek 62.: Vývoj populace rozpouštějících se smyček

6 Simulátor 2D celulárního automatu

V rámci této práce jsem vytvořil simulátor 2D celulárního automatu, který je schopen demonstrovat výsledky této práce. Simulátor je krom běžné simulace, která je samozřejmostí, schopen také vytvářet pravidla simulace, tato pravidla modifikovat, vytvářet grafické nadhledy nad pravidly, krokovat simulaci a různé pokročilejší funkce usnadňující experimentování se simulací 2D celulárních struktur. Simulator je vybaven systémem, který umožňuje přechod mezi jednotlivými sadami pravidel na základě globální informace o vývoji simulace. Tato pravidla je možné porovnávat a zjišťovat tak odchylky od původní sady pravidel. Simulátor umožňuje ukládat práci jako celek, nebo ukládat části práce tak, aby bylo možno tyto části libovolně používat v dalších experimentech nezávisle na původní zamýšlené simulaci. V tomto simulátoru je zabudován modul genetického algoritmu, který umožňuje evoluční návrh pravidel. Obrázek 63 zachycuje blokové schéma simulátoru.

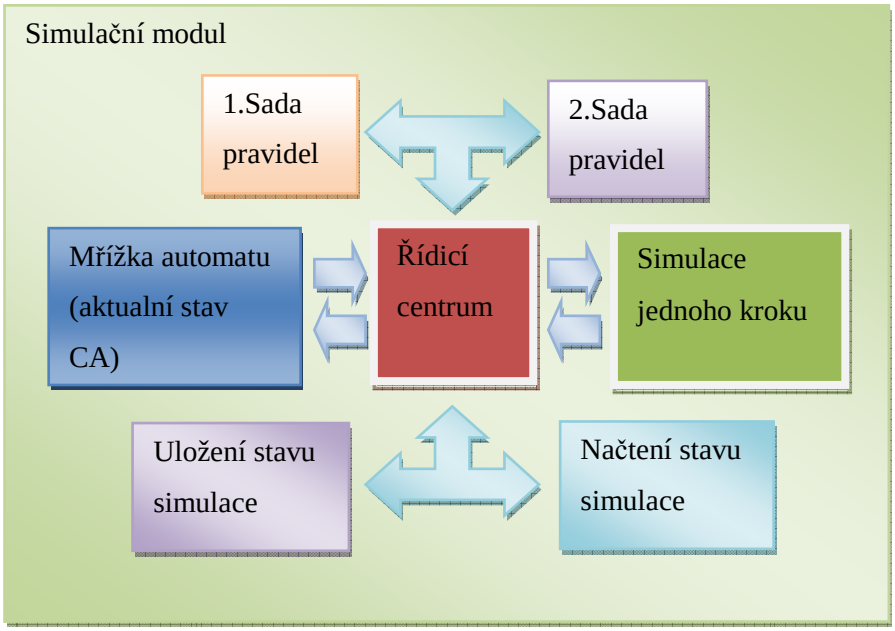


Obrázek 63.: Blokové schéma simulátoru

Jádrem celého systému je řídicí centrum, které komunikuje s ostatními částmi systémů, řídí průběh simulace i sestavování pravidel a inicializací. Uživatel ovládá simulátor přes grafické uživatelské rozhraní. Vizualizační modul zobrazuje výsledky simulace v přehledné grafice dvojrozměrné mřížky. Modul návrhu pravidel umožňuje navrhovat pravidla před nebo za běhu simulace a přímo tak simulaci ovlivňovat. Modul genetického algoritmu slouží k evolučnímu návrhu pravidel simulátoru. Tento modul bude podrobněji popsán v dalších částech práce. Samotnou simulaci pak provádí simulační modul. Tento modul může mít až dvě sady pravidel a podle pokynů řídicího centra provádí simulaci. Blokové schéma simulace je na následujícím obrázku 64.

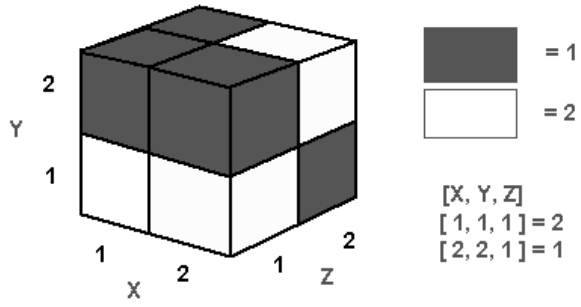
Simulace může být velice zdlouhavá. Pokud bychom uvažovali práci s von Neumannovo okolí a maximálně 10 stavů, může být počet všech pravidel 10^5. Tato pravidla se budou prohledávat pro každou buňku. Při nevhodném návrhu simulátoru může tedy nastat situace, že například pro mřížku 100x100 buněk, budeme 10000 krát prohledávat 1000000 pravidel pro jeden krok simulace. Tento simulátor řeší tento problém vytvořením multidimenzionální kostky (vícerozměrné pole), kde adresou je stav vyšetřované buňky a jejího okolí. Odpadává tak náročné prohledávání pravidel. Rychlost simulace je tedy úměrná počtu buněk mřížky. Nezáleží na počtu uvažovaných stavů nebo počtu pravidel, která se použijí v celulárním automatu. Rychlost simulace na mřížce 100x100 buněk

je stejná pro jednoduché binární struktury s XOR pravidlem tak i pro Langtonovu smyčku nebo Evoloop. Pokud počet buněk vzroste například desetkrát, desetkrát vzroste i doba simulace. Doba vyhodnocení následujícího stavu buňky je vždy konstantní a je provedena vždy v jednom kroku.



Obrázek 64.: Simulační modul

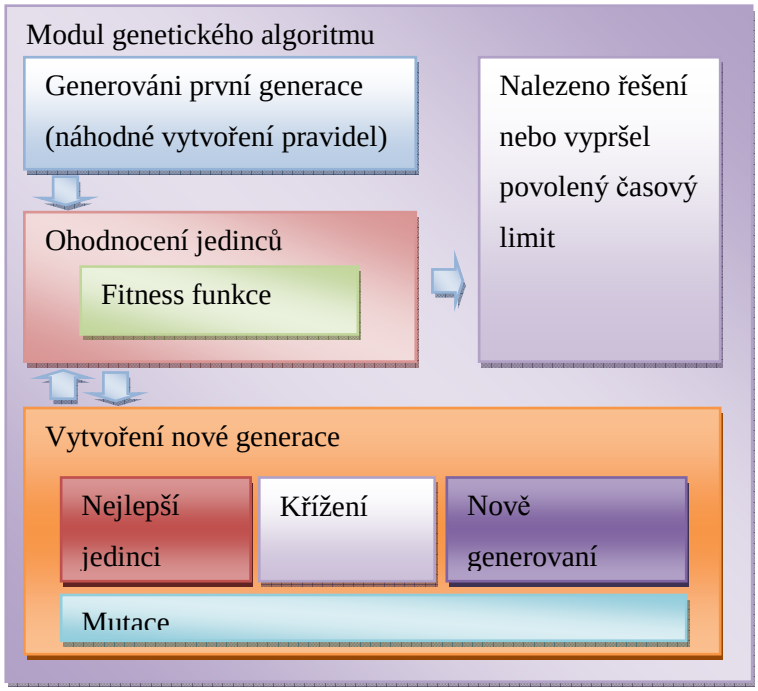
Na obrázku 65 je znázorněn princip struktury pravidel pro rychlou simulaci. Pro zjednodušení je jako příklad uvedena třírozměrná kostka. Tento automat může nabývat pouze dvou stavů. Celkový počet pravidel je maximálně osm. Z obrázku je dobře vidět, že při hledání pravidla zadáváme pouze adresu a hned známe výsledný stav vyšetřované buňky. Výsledek je tedy vždy nalezen v jednom kroku.



Obrázek 65.: Multidimenzionální kostka s pravidly

7 Evoluční návrh pravidel

V rámci této práce jsem se rozhodnul vyzkoušet evoluční návrh pravidel celulárního automatu. Využil jsem k tomuto účelu genetický algoritmus. Evoluční návrh pravidel 2D celulárního automatu je poměrně náročná věc. Jedná se především o ohodnocení kandidátních jedinců v generaci. Nad každým jedincem je nutné provést simulaci a výsledek ohodnotit, to může být časově velmi náročné a to také vytváří nároky na rychlou výpočetní techniku. Dalším problémem je velký stavový prostor všech možných pravidel. Následující blokové schéma (Obrázek 66) popisuje implementovaný genetický algoritmus.



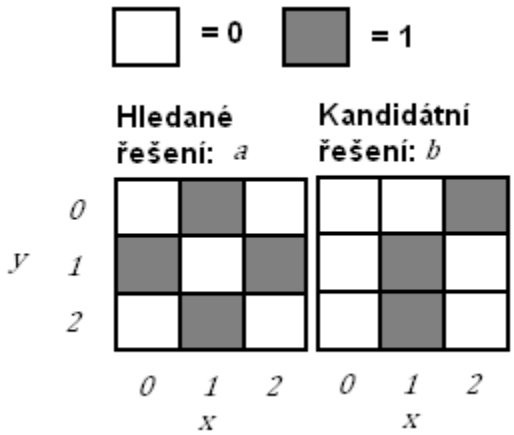
Obrázek 66.: Genetický algoritmus

V první fázi je vygenerována počáteční populace. Velikost populace nastaví uživatel před spuštěním algoritmu, stejně jako ostatní parametry. První populace je vygenerována náhodně. Jedinec je sada pravidel celulárního automatu.

Poté následuje ohodnocení populace. Tato část je stěžením prvkem celého algoritmu. Aby bylo možno jedince ohodnotit, je nutné spustit simulaci. V této fázi se tedy začíná spolupracovat se simulačním modulem. Je mu předán soubor pravidel a počáteční inicializace automatu. Simulace pak může probíhat ve dvou režimech. V prvním neznáme počet kroků, které musí simulátor vykonat pro dosažení cíle. V tomto případě je nutné ohodnotit pomocí fitness funkce výsledek každého kroku a pamatovat si nejlepší hodnotu. Také je nutné určit maximální počet kroků, které může simulátor vykonat, jinak by se simulace nikdy nezastavila. V druhém případě, kdy známe počet kroků, je situace mnohem jednoduší. Simulujeme předem stanovený počet kroků a na závěr ohodnotíme výsledek fitness funkcí. Tyto operace se provádí pro všechny jedince v populaci. Pokud jsme již našli

ideálního jedince, ukončíme celý algoritmus. V opačném případě pokračujeme v generování další generace.

Samotné hodnocení výsledků je velmi problematické. V tomto případě jsem volil jednoduchou chybovou funkci, ale bylo by možné použít například neuronovou síť. Chybová funkce porovná výslednou mřížku kandidátního jedince s mřížkou hledaného řešení (Obrázek 67) a vypočte sumu rozdílu stavů v odpovídajících si souřadnicích. Ideální jedinec je tedy ohodnocen nulou.



Obrázek 67.: Fitness funkce

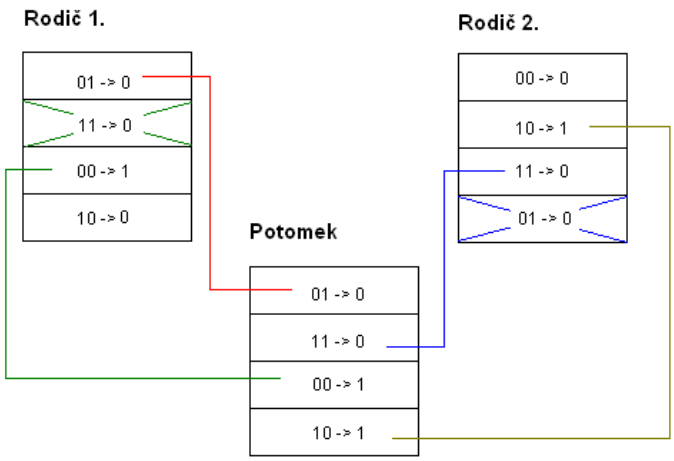
$$\sum |a_{xy} - b_{xy}| = |0-0| + |1-0| + |0-1| + |1-0| + |0-1| + |1-0| + |0-0| + |1-1| + |0-0| = 5$$

Rovnice 1.: Výpočet fitness

Pokud není nalezeno hledané řešení, je vytvořena nová generace. Vytváření je závislé na nastavení parametrů uživatelem. Do nové generace mohou postoupit nejlepší jedinci z generace předcházející. Vždy vzniknou noví jedinci křížením kvalitních jedinců z předchozí populace. Také můžeme část nové generace generovat náhodně jako nové jedince. V poslední řadě se pak uplatňuje generátor mutace, který náhodně provede změnu části populace.

Křížení jedinců probíhá následovně. Náhodně se vyberou rodiče nového jedince. Vybírají se z poloviny předchozí generace, která obsahuje kvalitnější jedince. Nový jedinec vznikne tak, že se vybírají postupně pravidla z jednoho rodiče a ukládají se do potomka. Poté se vezme pravidlo z druhého rodiče (začíná se brát od konce), zkontroluje se, jestli již potomek toto pravidlo v jisté formě neobsahuje (okolí je stejné, výsledný stav je jiný), a pokud ne, přidá se do nového jedince. Takto se tento proces opakuje, dokud není nový jedinec naplněn dostatečným množstvím pravidel. Princip je zachycen na obrázku Obrázek 68. Jde vlastně o druh jednobodového křížení.

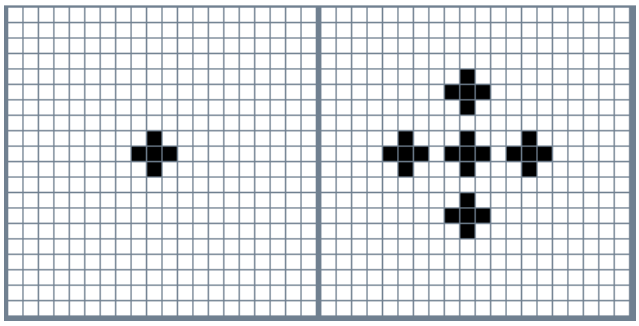
Mutace je proces, kdy se náhodně vybere jedinec (několik jedinců) a je pozměněn. Rozsah změn volí uživatel pomocí nastavení parametrů.



Obrázek 68.: Křížení jedinců

7.1 Výsledky evoluce pro jednoduchou strukturu, XOR pravidlo

Pokusil jsem se, aby evoluce navrhla pravidla jednoduché replikující se struktury popsané v předchozí části (kapitola 4.2). Počáteční inicializace automatu a stav, kterého chci pomocí vygenerovaných pravidel dosáhnout, jsou zachyceny na následujících obrázcích 69.



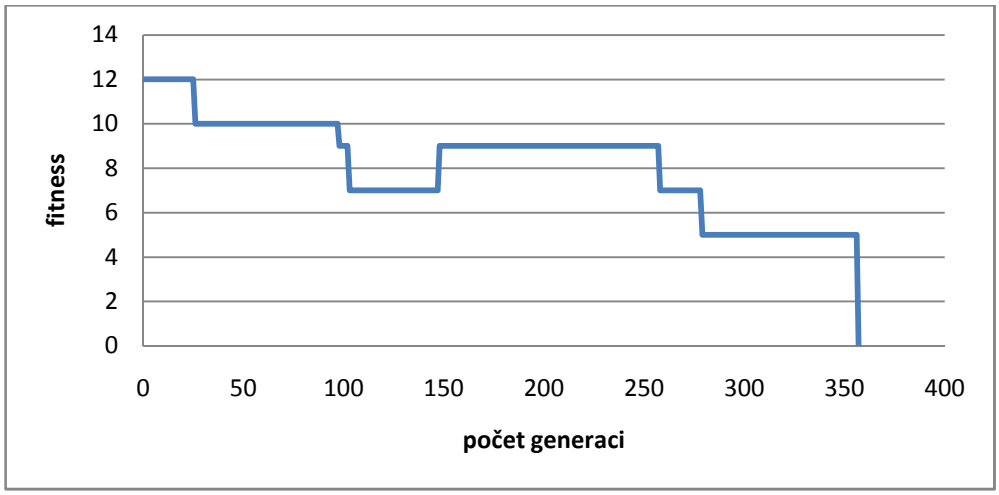
Obrázek 69.: Inicializace a cíl evoluce

Z obrázku je patrné, že struktura vytvoří čtyři své repliky a sama zůstane zachována. Repliky jsou vytvořeny ve čtyřech krocích. Jelikož se jedná o binární automat, je teoreticky stavový prostor všech sad pravidel 2^{2^5}. Celkem je potřeba navrhnout dvanáct pravidel, které v jednom kroku mění stav buňky. Tento fakt zmenšuje stavový prostor možných kandidátů.

Nejlepšího výsledku jsem dosáhl s následovně nastavenými parametry. Maximální počet pravidel jsem nastavil na 12 (pravidel, které mění stav buňky). Počet jedinců v generaci byl nastaven na 1000. Do nové generace vstoupilo 20% nejlepších jedinců z generace předchozí, 60% jedinců vzniklo křížením a 20% jich bylo náhodně generováno. Mutace probíhala maximálně u 5% populace.

V 357. generaci byl nalezen požadovaný výsledek. Následující graf (Obrázek 70) znázorňuje vývoj fitness nejlepšího jedince z generace. Z grafu je možné pozorovat, že ve 148. generaci

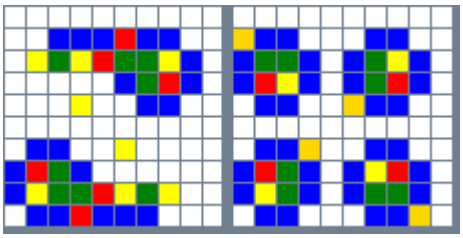
zmutoval nejúspěšnější jedinec v méně výhodného a vývoj se spíše zbrzdil, nicméně evoluce pokročovala úspěšně až k nalezení ideálního řešení.



Obrázek 70.: Vývoj fitness hodnoty nejlepšího jedince populace

7.2 Výsledky evoluce pro pravidla Bylovy smyčky

Také jsem se pokusil, aby evoluce navrhla pravidla Bylovy smyčky popsané v předchozí části (kapitola 4.5). Pomocí evoluce jsem chtěl dosáhnout pravidel, která urychlí replikaci smyčky. Doufal jsem v nalezení pravidel, která jsem sám navrhl, popřípadě v jejich lepší varianty. Počáteční inicializace automatu a stav, kterého chci pomocí vygenerovaných pravidel dosáhnout, jsou zachyceny na následujících obrázcích 71.



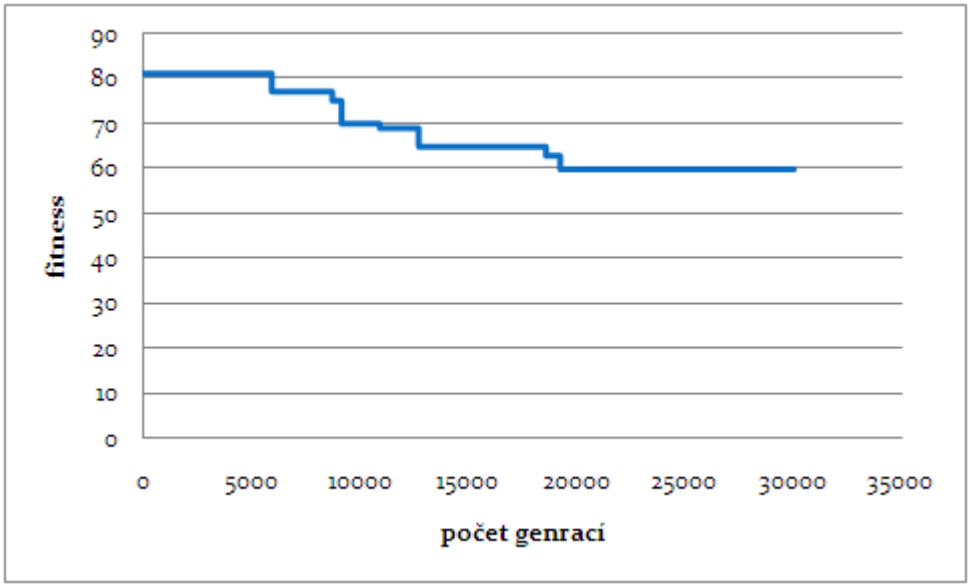
Obrázek 71.: Inicializace a cíl evoluce

V tomto případě je počáteční inicializací stav replikace v devátém kroku podle původních pravidel Bylovy smyčky. Moji snahou je najít nová pravidla, která by z tohoto stavu dokončila replikaci smyčky ve čtyřech krocích a tím zkrátila původní replikační dobu z pětadvaceti kroků na třináct. Ideou je urychlení replikace při změně pravidel na základě globální informace. Poznamenejme, že stavový prostor všech sad pravidel se oproti předcházejícímu experimentu značně zvětšil z $2^{2^5}+1\,na\,6^{6^5}+1$. Díky omezení počtu pravidel, které změní stav buňky, se stavový prostor podstatně zmenší.

Evoluce, oproti předcházejícímu experimentu (7.1), již tak úspěšná nebyla. Prováděl jsem pokusy s různým nastavením parametrů evoluce. V reálném čase (cca 20 hodin běhu algoritmu) jsem však nedosáhl požadovaných výsledků, i když se výsledky fitness funkce zlepšovaly a hodnoty

klesaly k nule. Byl by zřejmě potřeba větší výpočetní výkon a mnohem delší časový interval. Poznamenejme, že pokusy jsem prováděl na PC s konfigurací Intel Core2 Duo 3Ghz, 4 GB RAM, WINXP.

Naměřené výsledky zachycuje následující graf (Obrázek 72). Jedná se o fitness nejlepšího jedince v každé generaci. Evoluce měla nastavené parametry následovně. Populace byla o velikosti 3000 jedinců. Bylo povoleno maximálně 400 pravidel (měnící stav buňky). Do nové generace vstoupilo 20% nejlepších jedinců z generace předchozí, 60% jedinců vzniklo křížením a 20% jich bylo náhodně generováno. Mutace probíhala maximálně u 5% populace. Evoluce byla ukončena v 30000. generaci, kdy vypršen maximální časový limit povolený pro výpočet.



Obrázek 72.: Vývoj fitness nejlepšího jedince

To, že evoluce nenalezla požadovaná pravidla, není překvapující. Je to především díky velkému stavovému prostoru potencionálních řešení. Dále by mohlo evoluci pomoci vylepšené vícebodové křížení a v neposlední řadě kvalitnější fitness funkce realizována například neuronovou sítí. Použití neuronové sítě jako fitness funkce by bylo vhodnější, pokud bychom přesně neznali výsledný tvar hledané struktury ani výsledné umístění, což v mém experimentu nenastalo. Z tohoto důvodu jsem raději volil chybovou funkci. Chybová funkce vrátí výsledek obecně rychleji než neuronová síť, což je pro simulaci tohoto typu také velmi důležité.

Protože však tato problematika není přímo tématem této diplomové práce, nebudu se jí dále zabývat. Záměrem tohoto experimentu bylo zjistit, jestli by evoluce nemohla navrhnout kvalitnější pravidla, která by ještě výrazněji urychlila sebereplikaci.

8 Závěr

Cílem této práce měla být studie týkající se celulárních systémů a jejich využití k simulaci sebereplikace a její následné urychlení. Ve studii jsem se pokusil podrobně přiblížit problematiku týkajících se celulárních automatů a aplikací využívajících tyto systémy. Také jsem popsal některé architektury navržené na základech celulárních systémů. Nastudoval jsem a zdokumentoval principy některých sebereplikujících se struktur dvojrozměrných celulárních automatů. Hlavním přínosem této práce je však urychlení sebereplikace Bylovy smyčky. Díky změně pravidel celulárního automatu v 9. kroku simulace se replikace z 25 kroků urychlí na 13 kroků. Jedná se 48% urychlení oproti původní verzi. Také se mně podařilo rozšířit tuto smyčku o schopnost uvolnit své místo novým smyčkám poté, co již není schopna reprodukce dalších potomků. Dosáhl jsem tak velmi zajímavého emergenčního chování, které vykazují kolonie těchto struktur. Vytvořil jsem simulátor 2D celulárního automatu, který je schopen demonstrovat výsledky této práce. Také jsem se pokusil o evoluční návrh pravidel Bylovy smyčky pomocí genetického algoritmu.

Snaha urychlit tímto způsobem sebereplikaci vychází z několika skutečností. Víme, že existují struktury, které jsou schopny provádět svoji replikaci a zároveň i užitečný výpočet nebo jinou užitečnou činnost. Dále víme, že tyto systémy se vyznačují masivním paralelismem, tudíž při použití tohoto výpočetního modelu na vhodné platformě můžeme dosáhnout opravdu velmi dobrých výsledků. Dalším podstatným faktorem je existence polymorfních hradel, které mění logickou funkci v závislosti na externím stimulu, například ve formě změny napájecího napětí, nebo teploty. Tyto hradla by mohla být použity ke tvorbě zařízení, které by mohlo fungovat na principu celulárního automatu a externí stimul by mohlo měnit pravidla, od kterých by se funkce tohoto zařízení odvíjela. Toto zařízení by přesně korespondovalo se simulátorem celulárního automatu, který provádí simulaci sebereplikujících se struktur a v jistém okamžiku pozmění pravidla celulárního automatu a urychlí tak replikaci. Takovéto zařízení by mohlo vést ke tvorbě efektivních výpočetních platforem.

Dalším vývojem této práce by mohlo být sestavení hardwarového zařízení z polymorfních hradel, které by demonstrovalo urychlenou sebereplikací. Dalším směrem vývoje by mohlo být urychlení sebereplikací složitějších struktur, které provádí užitečné výpočty.

Samotná studie těchto systémů mě velmi nadchla a obohatila o velké množství nových poznatků. Chování těchto systémů je velmi fascinující a věřím, že pokud se bude pokračovat v jejich výzkumu, v budoucnu budou tyto systémy hrát velmi významnou roli v mnoha odvětvích vědy a techniky.

Literatura

[1] Csontó, J. aj.: *Umělý život*. In: Umělá inteligence 3, Praha, Academia, 2001, s. 76- 116, ISBN 80-200-0472-6

[2] Sayama, H.: *Constructing evolutionary systems on a simple deterministic cellular automata space*. [Ph.D. Dissertation]. University of Tokyo, Department of Information Science, Graduate School of Science, 1998

[3] Husáková, M.: *Celulálární automaty*. [Studijní matriál]. Dokument dostupný na URL http://lide.uhk.cz/fim/ucitel/fshusam2/lekarnicky/zt3/zt3_dokumenty/CelularniAutomaty (prosinec 2008)

[4] Durbeck, L., Macias, N.: *The Cell Matrix: an architecture for nanocomputing.* In Nanotechnology 12, Cell Matrix Corporation, Salt Lake City, USA, s. 217-230, 2001

[5] Wolfram, S.: *Cellular automata: The Nature of Cellular Automata and a Simple Example.* In: Los Alamos Science, s. 2-21, 1983, Dokument dostupný na URL http://www.stephenwolfram.com/publications/articles/ca/83-cellular/2/text.html (prosinec 2008)

[6] Sekanina, L.: *Development v evolučním návrhu.* [Studijní materiál]. Vysoké učení technické v Brně, Fakulta informačních technologií, Božetěchova 2, 612 66 Brno, 2008

[7] Sekanina, L.: *Embryonální hárdware.* [Studijní materiál]. Vysoké učení technické v Brně, Fakulta informačních technologií, Božetěchova 2, 612 66 Brno, 2008

[8] Žaloudek, L.: *Sebereplikace ve výpočetních systémech*, In: Počítačové architektury a diagnostika 2008, Liberec, CZ, TUL, 2008, s. 131-136, ISBN 978-80-7372-378-1

[9] Zykov, V. aj.: *Self-reproducing machine.* In: Nature, vol. 435, 2005, s. 163, Dokument dostupný na URL http://ccsl.mae.cornell.edu/papers/Nature05_Zykov.pdf (leden 2009)

[10] *Langton's loop.* Wikipedia, the free encyklopedia Dokument dostupný na URL http://en.wikipedia.org/wiki/Langton%27s_loops#cite_ref-Tempesti1995_5-0 (březen 2009)

[11] Oros, N., Nehaniv, C, L.: *Sexyloop: Self-Reproduction, Evolution and Sex in Cellular Automat* In: Artificial Life, The First IEEE Symposium on Artificial Life, Hawaii IEEE, USA, 2007, s. 130-137

[12] Oros, N., *Sexyloop: Self-Reproduction, Evolution and Sex in Cellular Automata.* Dokument dostupný na http://homepages.feis.herts.ac.uk/~on5ag/sexyloop.html (březen 2009)

[13] *Codd's cellular automata.* Wikipedia, the free encyklopedia Dokument dostupný na URL http://en.wikipedia.org/wiki/Codd%27s_cellular_automaton (březen 2009)

[14] Sipper, M.: *Accompanying Figures.* In: The Self-Replication Page, Logic Systems Laboratory, Swiss Federal Institute of Technology, CH-1015 Lausanne, Switzerland, 1997, Dokument dostupný na URL http://www.cs.bgu.ac.il/~sipper/selfrep/figures.pdf

[15] Tampesti, G.: *A Self-Repairing Multiplexer-Based FPGA Inspired by Biological Processes.* Dokument dostupný na URL http://carg2.epfl.ch/Teaching/GDCA/loops-thesis.pdf (březen 2009)

[16] Byl, J.: *Self-Reproduction in small cellular automata.* In: Physica D, vol. 34, 1989, s. 295-299

[17] Reggia, J., A., Armentrout, S.,L., Chou, H., Peng, Y.: *Simple systems that exhibit self-directed replication.* In Science, vol. 259, 1993, s. 1282-1287

[18] Antony, A., Gavidia, D., Salzberg, Ch.: *Self-Reproducing Cellular Automata: Introducing Memory in Evoloops*. Section Computational Science, Universiteit van Amsterdam, 2002, Dokument dostupný na URL http://staff.science.uva.nl/~sloot/CSS/memloop.pdf (květen 2009)

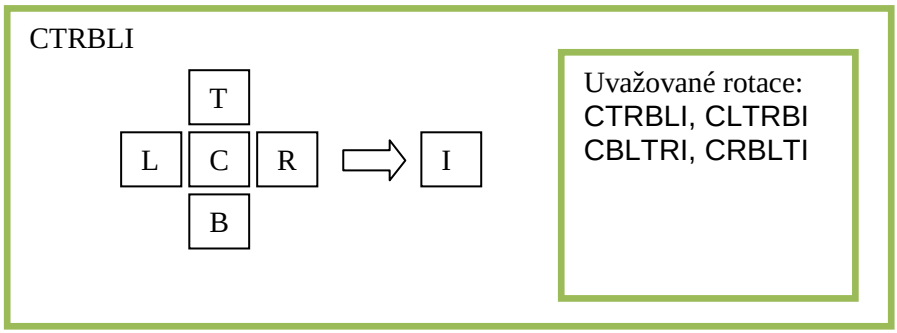
[19] Dariol, D.: *A Self-Replication.* In: Artificial Life, Institut für Informatik der Universität Zürich, 2001, Dokument dostupný na URL http://www.ifi.uzh.ch/ailab/teaching/AL01/chap7.doc (květen 2009)

[20] Sekanina L. aj.: *Polymorphic Gates in Design and Test of Digital Circuits*. In: International Journal of Unconventional Computing, roč. 4, č. 2, Philadelphia, USA, 2008, s. 125-142, ISSN 1548-7199

Seznam příloh

Příloha 1. Původní pravidla Byl's loop

V této části jsou uvedeny pravidla pro původní Bylovu smyčku. Pravidla jsou uvedena včetně rotací pro funkci replikace ve všech směrech. Celkový počet pravidel je 238. Všechna pravidla jsou uvedena ve tvaru CTRBLI , kde jednotlivé znaky symbolizují pozice uvedené na následujícím schématu.



CTRBLI	CTRBLI	CTRBLI	CTRBLI	CTRBLI	CTRBLI	CTRBLI
000031	100033	132414	251005	313321	400233	452313
000135	100100	132434	251205	315121	401233	453123
000212	100303	133000	252025	315325	402303	500132
000233	100330	133244	253203	320512	402520	500242
000242	101000	134424	300010	321001	405235	500422
000301	101233	135423	300030	321011	411233	501302
000311	101244	140124	300100	321031	412233	502124
000512	103003	140324	300110	321121	412303	502220
000525	103244	141124	300211	321131	412313	502230
001305	103300	141224	300300	321321	412533	502402
002102	110000	141324	301000	321331	414323	504202
002303	111244	142344	301100	321511	415233	504422
002402	112244	142353	301211	321535	420250	512024
003001	112303	143224	302101	322111	421433	513002
003101	112404	143324	303000	322131	422313	520042
005102	112414	144234	303211	323411	423003	520214
005205	113244	154233	305122	325131	423013	520220
010022	122414	200000	310000	325415	423055	520442
010031	122434	200220	310010	330000	423113	521204
010052	123013	200515	310021	332101	423123	522020
013005	123444	202200	310121	332111	423153	522200
020055	123543	202525	310321	332131	425200	522300
021002	124014	205105	311000	332155	425313	523020
023003	124034	205125	311221	332211	430023	523242
024002	124114	205323	311321	332511	430123	524002
030001	124124	210055	312052	333211	430525	524232
030015	124134	212055	312101	334121	431123	530012
030023	124324	220020	312151	341231	431223	530220
031001	124334	220255	312211	341255	431253	532422
040022	130003	220515	312341	351202	431523	540022
051002	130030	220533	312545	351211	432143	542002
052005	130123	222000	313211	351321	443213	542042
100000	132244	225205	313221	353215	452020	542322
100010	132404	232053	313251	354125	452305	544202

Příloha 2. Nová pravidla Byl’s loop (pro 10.-13. krok replikace)

Jedná se o 277 pravidel, do kterých jsou již zahrnuty rotace pro funkci replikace ve všech směrech.

CTRBLI	CTRBLI	CTRBLI	CTRBLI	CTRBLI	CTRBLI	CTRBLI
000015	030001	124134	221200	322111	423113	504422
000031	030015	124334	222000	323411	423153	505000
000105	030025	130003	225205	324432	423200	510022
000135	031001	130030	242040	325131	425100	510034
000212	040025	130123	244200	325415	425200	511502
000235	040042	130452	251005	330000	425313	512024
000245	040052	132414	251205	332101	430002	513002
000301	042005	132434	252025	332155	430023	515012
000311	044002	133000	300010	332442	430525	520042
000425	045002	133244	300030	332511	431123	520214
000442	050042	134424	300100	334121	431253	520220
000452	051002	135423	300110	341231	431523	520442
000512	052205	140124	300211	341255	432020	521002
000542	054002	141124	300300	343242	432143	521204
001005	100000	141224	301000	344322	443213	522020
001015	100010	141324	301100	351202	451020	522200
001305	100033	142344	302101	351211	452020	522300
002102	100100	142353	303000	351321	452305	523020
002305	100303	143324	303211	353215	452313	523242
002405	100330	144234	305122	354125	453123	524002
003001	101000	145302	310000	400032	500000	524232
003101	101233	153042	310010	400233	500034	525540
004205	101244	154233	310021	400302	500050	530004
004402	103003	200000	310321	402303	500132	530012
004502	103300	200220	311000	402320	500212	530220
005102	104532	200515	311221	402510	500242	531004
005225	110000	202120	312052	402520	500304	532422
005402	111244	202200	312151	403002	500314	540022
010005	112244	202525	312211	405235	500422	542002
010022	112303	204420	312341	410250	500500	542042
010031	112404	205105	312545	411233	501152	542322
010052	112414	205125	313251	412313	501302	542550
010105	113244	210055	315121	412533	502102	544202
013005	122414	212020	315325	414323	502124	550000
020045	123013	212055	320512	415233	502220	550112
020525	123444	220020	321001	420230	502230	554250
021002	123543	220210	321031	420250	502402	555420
022055	124014	220255	321121	421433	503004	
023005	124114	220440	321511	423003	503104	
024005	124124	220515	321535	423055	504202	

Příloha 3. Rozdíly mezi původními a novými pravidly Byl's loop

Jedná se o sjednocení dvou sad pravidel a vyznačení průniku levých stran pravidel, u kterých došlo ke změně v pravých stranách. Celkem tímto sjednocením vznikne 317 pravidel a 127 průniků se změnou. Tedy jedná se o 40. 06309% změnu. Pravidla jsou uvedena ve tvaru CTRBLI >V kde „V" označuje výsledek levé strany pravidla pro novou sadu pravidel. „I" značí výsledek pravidla pro původní sadu. Pokud je za tímto tvarem uveden znak „!" znamená to, že došlo ke změně. Pravidla jsou uvedena včetně rotací pro funkci replikace ve všech směrech. Změny jsou zvýrazněny červeně.

CTRBLI>V	CTRBLI>V	CTRBLI>V	CTRBLI>V	CTRBLI>V	CTRBLI>V	CTRBLI>V
000010>5!	031001>1	130123>3	242042>0!	321331>3!	420234>0!	502402>2
000031>1	040022>5!	130451>2!	244202>0!	321511>1	420250>0	503005>4!
000100>5!	040040>2!	132244>1!	251005>5	321535>5	421433>3	503105>4!
000135>5	040050>2!	132404>1!	251205>5	322111>1	422313>4!	504202>2
000212>2	042000>5!	132414>4	252025>5	322131>3!	423003>3	504422>2
000233>5!	044000>2!	132434>4	253203>2!	323411>1	423013>4!	505005>0!
000242>5!	045000>2!	133000>0	300010>0	324433>2!	423055>5	510025>2!
000301>1	050040>2!	133244>4	300030>0	325131>1	423113>3	510035>4!
000311>1	051002>2	134424>4	300100>0	325415>5	423123>4!	511505>2!
000420>5!	052005>0!	135423>3	300110>0	330000>0	423153>3	512024>4
000440>2!	052200>5!	140124>4	300211>1	332101>1	423204>0!	513002>2
000450>2!	054000>2!	140324>1!	300300>0	332111>3!	425104>0!	515015>2!
000512>2	100000>0	141124>4	301000>0	332131>3!	425200>0	520042>2
000525>0!	100010>0	141224>4	301100>0	332155>5	425313>3	520214>4
000540>2!	100033>3	141324>4	301211>3!	332211>3!	430004>2!	520220>0
001000>5!	100100>0	142344>4	302101>1	332443>2!	430023>3	520442>2
001010>5!	100303>3	142353>3	303000>0	332511>1	430123>4!	521005>2!
001305>5	100330>0	143224>1!	303211>1	333211>3!	430525>5	521204>4
002102>2	101000>0	143324>4	305122>2	334121>1	431123>3	522020>0
002303>5!	101233>3	144234>4	310000>0	341231>1	431223>4!	522200>0
002402>5!	101244>4	145301>2!	310010>0	341255>5	431253>3	522300>0
003001>1	103003>3	153041>2!	310021>1	343243>2!	431523>3	523020>0
003101>1	103244>1!	154233>3	310121>3!	344323>2!	432024>0!	523242>2
004200>5!	103300>0	200000>0	310321>1	351202>2	432143>3	524002>2
004400>2!	104531>2!	200220>0	311000>0	351211>1	443213>3	524232>2
004500>2!	110000>0	200515>5	311221>1	351321>1	451024>0!	525545>0!
005102>2	111244>4	202122>0!	311321>3!	353215>5	452020>0	530005>4!
005205>0!	112244>4	202200>0	312052>2	354125>5	452305>5	530012>2
005220>5!	112303>3	202525>5	312101>3!	400034>2!	452313>3	530220>0
005400>2!	112404>4	204422>0!	312151>1	400233>3	453123>3	531005>4!
010000>5!	112414>4	205105>5	312211>1	400304>2!	500005>0!	532422>2
010022>2	113244>4	205125>5	312341>1	401233>4!	500035>4!	540022>2
010031>1	122414>4	205323>2!	312545>5	402303>3	500055>0!	542002>2
010052>2	122434>1!	210055>5	313211>3!	402324>0!	500132>2	542042>2
010100>5!	123013>3	212022>0!	313221>3!	402514>0!	500215>2!	542322>2
013005>5	123444>4	212055>5	313251>1	402520>0	500242>2	542555>0!
020040>5!	123543>3	220020>0	313321>3!	403004>2!	500305>4!	544202>2

020055>0!	124014>4	220212>0!	315121>1	405235>5	500315>4!	550005>0!
020520>5!	124034>1!	220255>5	315325>5	410254>0!	500422>2	550115>2!
021002>2	124114>4	220442>0!	320512>2	411233>3	500505>0!	554255>0!
022050>5!	124124>4	220515>5	321001>1	412233>4!	501155>2!	555425>0!
023003>5!	124134>4	220533>2!	321011>3!	412303>4!	501302>2	
024002>5!	124324>1!	221202>0!	321031>1	412313>3	502105>2!	
030001>1	124334>4	222000>0	321121>1	412533>3	502124>4	
030015>5	130003>3	225205>5	321131>3!	414323>3	502220>0	
030023>5!	130030>0	232053>2!	321321>3!	415233>3	502230>0	

Příloha 4. První sada pravidel pro rozpouštějící se Byl’s loop

V tabulce jsou uvedena pravidla, která je nutno přidat k původní sadě pravidel Bylovy smyčky. Pravidla jsou uvedena včetně rotací pro funkci replikace ve všech směrech.

CTRBLI	CTRBLI	CTRBLI	CTRBLI	CTRBLI	CTRBLI	CTRBLI
600000						
600665						
606605						
606665						
660065						
660665						
666005						
666065						
666605						
666665						

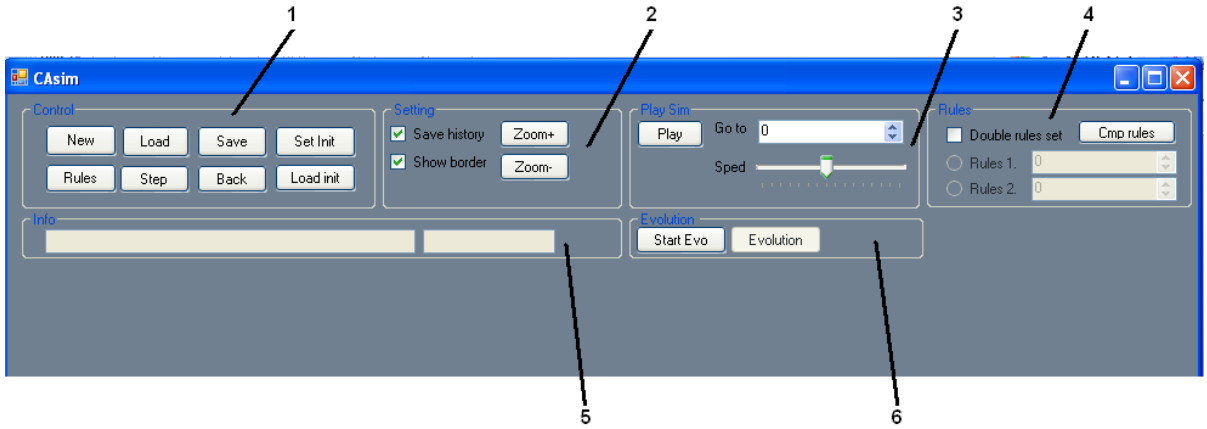
Příloha 5. Druhá sada pravidel pro rozpouštějící se Byl’s loop

V tabulce jsou uvedena pravidla, která je nutno přidat k nové (druhé) sadě pravidel Bylovy smyčky. Pravidla jsou uvedena včetně rotací pro funkci replikace ve všech směrech.

CTRBLI	CTRBLI	CTRBLI	CTRBLI	CTRBLI	CTRBLI	CTRBLI
124336	250266	355332	505250	553506		
132436	252250	362266	505356	553600		
133246	255220	363226	505360	555554		
143326	256206	366226	506200	560020		
200266	260026	422366	506350	560060		
200626	260066	423626	506600	560530		
200666	262006	436226	520050	562000		
202606	262056	444553	520060	563500		
202656	262260	445543	525000	566000		
205626	265026	454453	525050	605250		
206206	266006	455443	526000	625050		
206606	266220	462236	535056	650520		
220066	322636	500250	535060	652500		
220566	322666	500260	536050	655555		
222550	326326	500520	550020			
222660	326626	500620	550520			
225520	332266	500660	550536			
226006	333552	502500	550630			
226506	335532	502600	552000			
226620	353352	505200	552500			

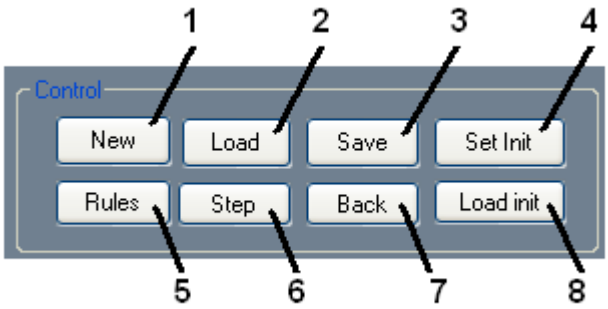
Příloha 6. Manuál k simulátoru

Program je implementován v jazyku C# v programu Microsoft Visual Studio 2005 (Framework 2.0). Simulátor můžeme spustit pomocí souboru casim.exe nebo přeložit a spustit z dodaných zdrojových kódů. Je nutné použít operační systém Windows (min. Framework 2.0). Po spuštění programu se zobrazí následující formulář. Poznamenejme, že tento simulátor uvažuje von Neumannovo okolí a cyklické okrajové podmínky.

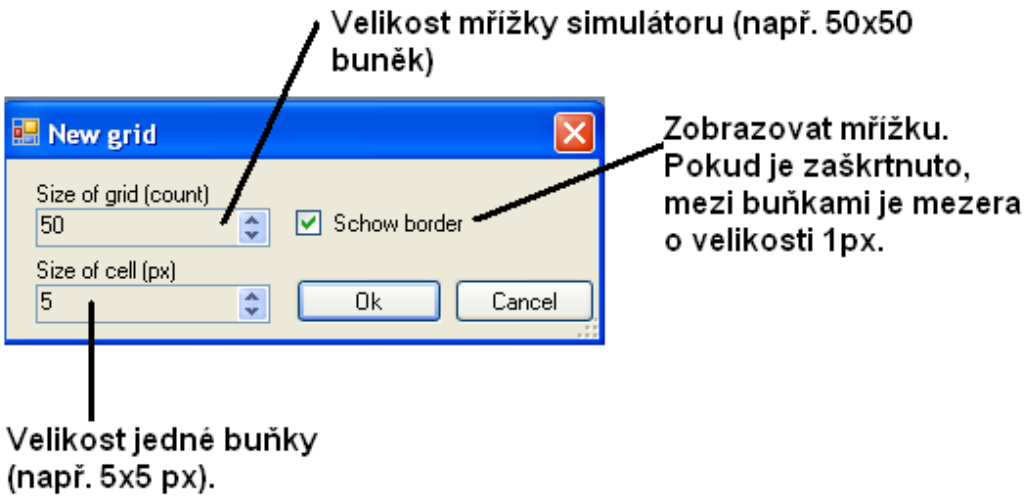


1. Panel s tlačítky pro ovládání simulace.
2. Panel, který ovládá zobrazování a velikost mřížky + ukládání historie simulace.
3. Panel pro ovládání plynulého chodu simulace.
4. Panel pro ovládání sad pravidel.
5. Panel, který vypisuje informace o poloze na mřížce.
6. Panel pro ovládání evolučního algoritmu.

Panel pro ovládání simulace

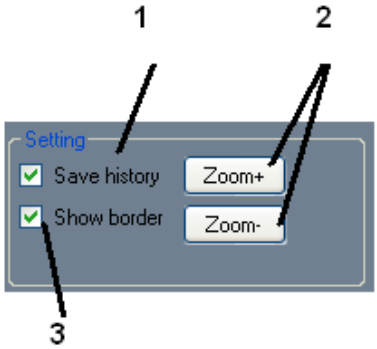


1. Tlačítko pro zahájení nového projektu. Zobrazí se následující formulář. Po nastavení údajů a stisku tlačítka *Ok* je vytvořen nový projekt a zobrazí se prázdná mřížka.



2. Načte uložený projekt (zobrazí se OpenDialog). Simulátor se zeptá, jestli má při načítání pravidel vytvářet i rotace uvedených pravidel (rotace jsou popsány v příloze 1.).
3. Uloží (zobrazí se SaveDialog) právě rozpracovaný projekt (sady pravidel, aktuální stav buněk automatu, případně kroky přepínání sad pravidel).
4. Při provádění experimentu se simulátorem je někdy potřeba vrátit simulaci do určitého stavu a z tohoto stavu různě experimentovat s jinak nastavenými pravidly. Tato volba uloží aktuální stav buněk a stav simulátoru jako bod, ke kterému je možno se vrátit (při načtení projektu pomocí *Load* je nastaven jako výchozí bod načtená inicializace automatu).
5. Tlačítko spouští formulář pro zprávu pravidel. Podrobně se tomuto formuláři budu věnovat dále.
6. Provede jeden krok simulace.
7. Pokud je povolená historie, je možné krokovat simulaci nazpět (do předchozích stavů buněk). Je možné provést pouze pět kroků zpět.
8. Vrátí stav simulace do počátečního stavu nebo do stavu nastaveného volbou popsanou v odrážce „4“.

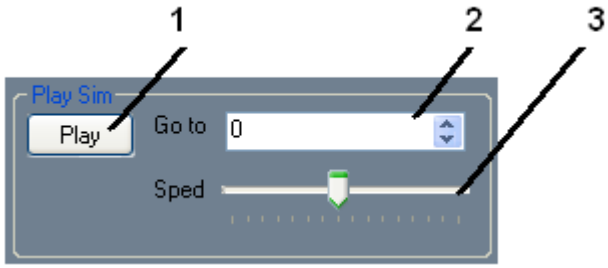
Panel pro ovládání zobrazení



1. Ukládá pět kroků historie simulace pro zpětné krokování.

2. Zvětšuje a zmenšuje buňky.
3. Pokud je zaškrtnut, zobrazuje mřížku (mezi buňkami je 1px rámečku).

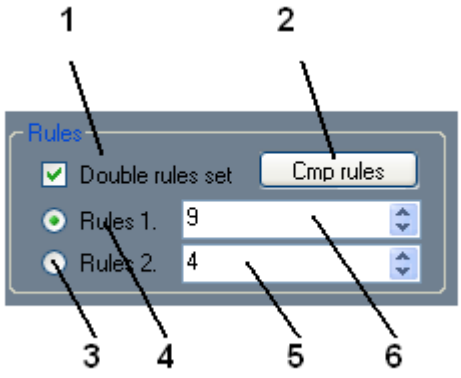
Panel pro ovládání plynulé simulace



1. Spustí se simulace bez nutnosti klikat na tlačítko *Step*. Pokud je v editu *Go to* nastavena nula, pokračuje simulace bez omezení do nekonečna. Pokud je v editu nastavena nenulová hodnota, tak se provede právě tolik kroků simulace.
2. Pokud je v editu *Go to* nastavena nula, pokračuje simulace do nekonečna. Pokud je v editu nastavena nenulová hodnota, tak se provede právě tolik kroků simulace.
3. Nastavuje rychlost simulace.

Panel pro ovládání sady pravidel

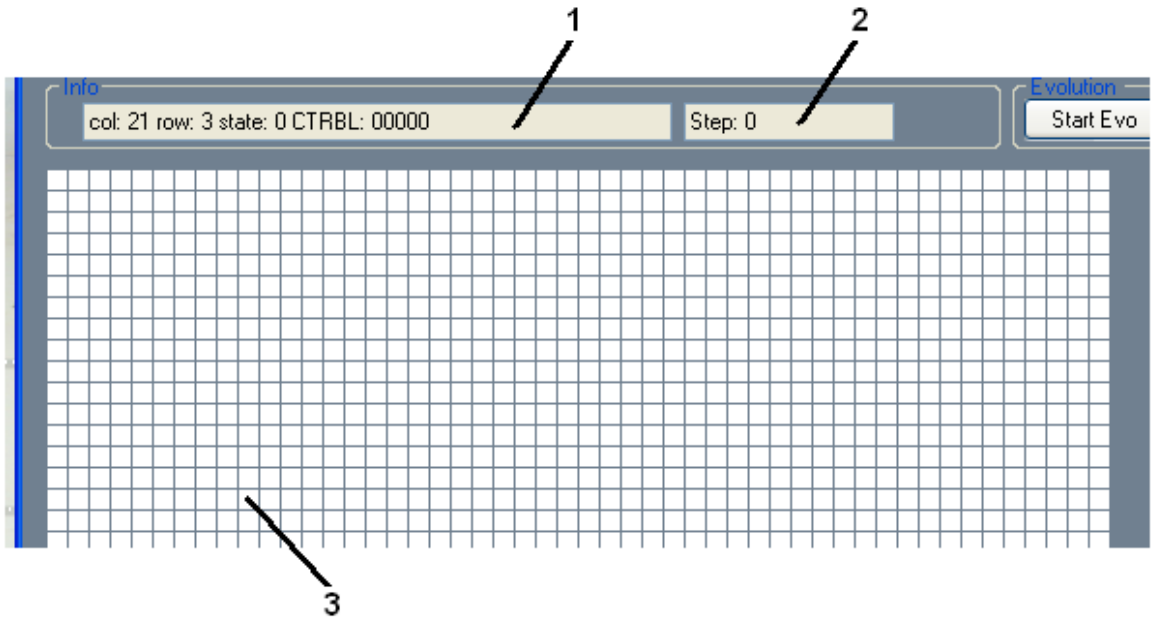
Simulátor muže použít dvě sady pravidel pro celulární automat. V tomto panelu se nastavuje, kdy se s kterou sadou pracuje.



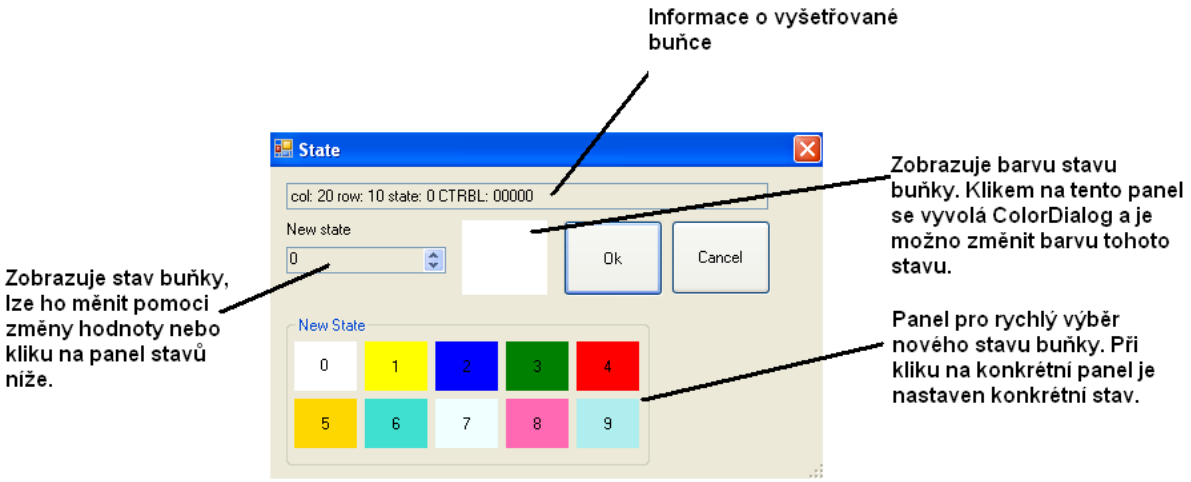
1. Pokud je zatrženo, pracuje se s oběma sadami pravidel. Pokud není zatrženo, pracuje se pouze s první sadou.
2. Zobrazí formulář s porovnáním obou sad pravidel.
3. Pokud je zatrženo, simulátor pracuje s druhou sadou pravidel.
4. Pokud je zatrženo, simulátor pracuje s první sadou pravidel.

5. Hodnota nastavuje počet kroků, které jsou s příslušnou sadou provedeny. Pak dojde k přepnutí na další sadu. Pokud je nastavena nula, sada pravidel se nepoužije.
6. Hodnota nastavuje počet kroků, které jsou s příslušnou sadou provedeny. Pak dojde k přepnutí na další sadu. Pokud je nastavena nula, sada pravidel se nepoužije.

Zobrazená mřížka (buňky) simulátoru a nastavování stavů jednotlivých buněk

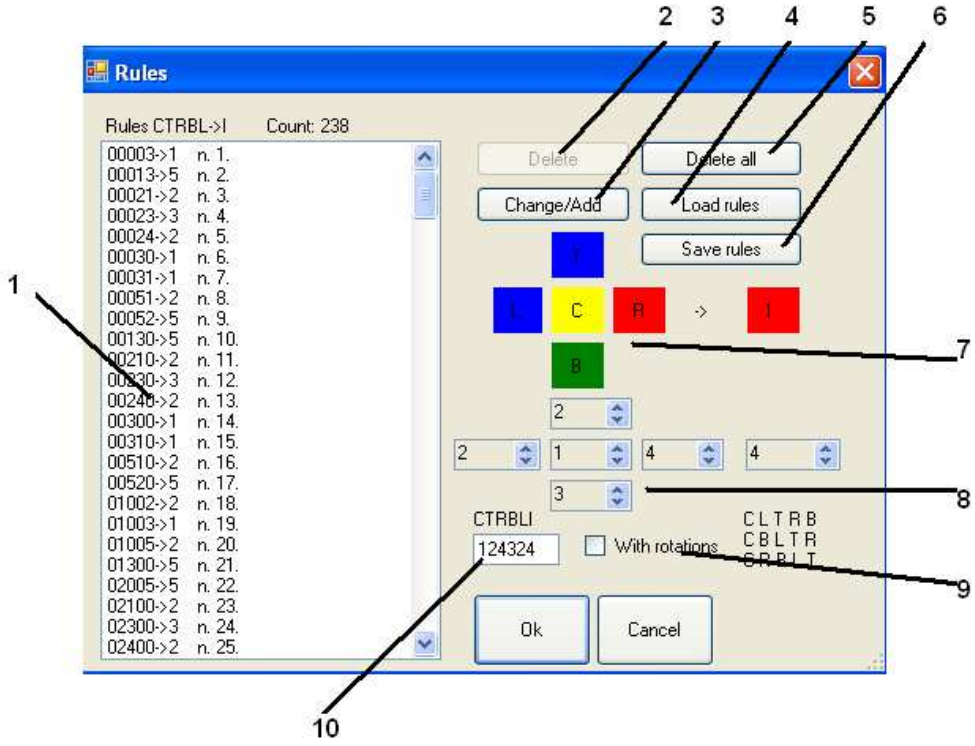


1. Zobrazuje informace o buňce, nad kterou se nachází kurzor.
2. Zobrazuje aktuální krok simulace.
3. Zobrazuje mřížku buněk automatu. Pokud chci nastavit konkrétní stav buňky, klikem levého tlačítka vyvolám formulář pro zadání stavu. Klik pravého tlačítka vyvolá nastavení pravidla přímo pro konkrétní situaci buňky a jejího okolí (více v další části). Následující obrázek popisuje formulář pro nastavování stavů.



Vytváření a změna pravidel

Simulátor pracuje až s dvěmi sady pravidel. Pracuje se vždy se sadou označenou v panelu pro ovládání sady pravidel. Pokud nejsou dvě sady povoleny, pracuje se s první sadou pravidel. Pokud chci vyvolat formulář pro správu pravidel, kliknu na tlačítko *Rules* nebo pokud chci zjistit pravidlo, které se použije v následujícím kroku simulace (pokud se ale v tomto kroku nezmění sady pravidel!) pro konkrétní buňku, kliknu pravým tlačítkem myši na příslušnou buňku a otevře se formulář s vyhledaným pravidlem a je možno toto pravidlo rovnou editovat.



1. Zobrazuje pravidla v aktuální sadě pravidel. Zobrazuje pouze pravidla, která mění stav buňky. Konkrétní pravidlo je možné vybrat a editovat pomocí označení (kliku pravého tlačítka myši).

Poté je toto pravidlo nastaveno do nástrojů pro úpravu pravidla označené na obrázku jako „7", „8", „10".

2. Vymaže označené pravidlo.
3. Potvrdí a provede změnu či přidání nového pravidla (musí se stisknout, nestačí tlačítko *Ok*).
4. Načte uložená pravidla. Načítá pouze pravidla do aktuální sady ze souboru s pravidly, nikoli celý projekt.
5. Vymaže všechny pravidla v sadě.
6. Uloží pravidla aktuální sady do souboru.
7. Nástroj pro editaci nebo vytváření nového pravidla. Při kliku na konkrétní buňku se zobrazí panel pro výběr nového stavu příslušné buňky. Výběr ovlivňuje i hodnoty v „8" a „10".
8. Nástroj pro editaci nebo vytváření nového pravidla. Výběr ovlivňuje i hodnoty v „7" a „10".
9. Pokud je zaškrtnuto, změna nebo uložení nového pravidla se neprojeví pouze na tomto pravidlu, ale i na jeho rotačních ekvivalentech (i na pravidlech CLTRB, CBLTR a CRBLT).
10. Nástroj pro editaci nebo vytváření nového pravidla. Změna ovlivňuje i hodnoty v „7" a „8".

Pokud chci vytvořit nové pravidlo, které ještě není uvedeno v seznamu pro konkrétní sadu, tak pomocí nástrojů pro editaci pravidla nastavím požadované hodnoty a zvolím tlačítko *Change/Add* (případně před stiskem tohoto tlačítka zvolím, jestli chci přidat i rotace tohoto pravidla).

Evoluční návrh pravidel

Evoluce je velmi zdlouhavý proces (často i několik hodin).

1. Pokud jsem se simulací nebo nastavením buněk dospěl do stavu, který je výchozí inicializací buněk automatu pro evoluci, stisknu toto tlačítko.
2. Pokud jsem se simulací nebo nastavením buněk dospěl do stavu, který je cílem, jehož chci pomocí evolučně hledaných pravidel dosáhnout, stisknu toto tlačítko a spustí se formulář pro nastavení parametrů evoluce (je velmi intuitivní, proto ho nebudu popisovat).

Příloha 7. Demonstrace výsledků

Následující postup popisuje, jak demonstrovat výsledky této práce. Doporučuji nejprve přečíst přílohu 6. pro získání představy o funkcích simulátoru. Uložené projekty pro demonstraci jsou v adresáři *savedata*. V tomto adresáři je pět souborů, které jsou popsány v následující tabulce.

Jméno souboru	Popis
Bl	Byl's loop pomocí jedné sady pravidel
Langton	Langtonova smyčka
Bfl	Byl's loop s urychlenou sebereplikací pomocí dvou sad pravidel
Bflsd	Byl's loop s urychlenou sebereplikací a rozpouštějící strukturou
bflsd185	Byl's loop s urychlenou sebereplikací a rozpouštějící strukturou ve 185. kroku simulace

1. Spusťte program casim.exe nebo přeložte a spusťte zdrojové kódy simulátoru.
2. Stiskněte tlačítko *Load*. Otevře se dialog pro výběr souboru. Zvolte jeden z výše popsaných souborů. Simulátor se Vás zeptá, jestli má vytvářet i rotace uvedených pravidel (*Rotation rules?*). Zvolte *Ne*, protože uložené soubory již rotace pravidel obsahují.
3. Projekt je načten. Spustit simulaci můžete pomocí tlačítka *Play* nebo krokovat simulaci pomocí tlačítka *Step*.
4. Pokud se dostanete se simulací do určité fáze a budete se chtít vrátit do původního stavu, doporučuji projekt znovu načíst nebo použít tlačítko *Load init* (v tomto případě dejte pozor na správnou sadu pravidel, která je právě nastavena a případně nastavte první sadu pravidel ručně).

Příloha 8. Seznam obrázků

Příloha 9. Seznam tabulek