SEBEREPLIKACE VE VÝPOČETNÍCH SYSTÉMECH

Luděk Žaloudek
Výpočetní technika a informatika, 1. ročník, prezenční studium
Školitel: Lukáš Sekanina

Fakulta informačních technologií, Vysoké učení technické v Brně
Božetěchova 2, 612 66 Brno

izaloude@fit.vutbr.cz

Abstrakt. Tento příspěvek se zabývá možnými přístupy k sebereplikaci v celulárních automatech, která se ukazuje jako zajímavý způsob řešení některých problémů u masivně paralelních výpočetních architektur. Dále je naznačen další směr výzkumu s přihlédnutím k disertační práci.

Klíčová slova. Celulární automaty, sebereplikace, masivně paralelní výpočetní architektury, univerzální výpočty, teorie poziční informace.

1 Úvod

V posledních několika letech se změnil trend ve způsobu zvyšování výkonu výpočetních systémů. Z důvodů vysokého ztrátového výkonu při vyšších frekvencích se návrháři takových systémů vydali cestou paralelizace. Namísto jednoho složitého procesoru obsahují dnešní výpočetní systémy stále větší počty stále jednodušších jader. Příkladem můžou být procesory IBM Cell či shluky procesorů na grafických kartách nVidia (CUDA).

Zatímco zmíněné systémy dnes obsahují obvykle desítky výpočetních jader, v budoucnu to budou stovky či tisíce. Největšími problémy těchto tzv. masivně paralelních výpočetních systémů jsou nutnost centrálního řízení a pomalá konfigurace a komunikace. Dalším nedostatkem může být nedostatečná odolnost proti poruchám. Z těchto důvodů se objevují nejrůznější výpočetní modely inspirované biologií, které by mohly znamenat řešení zmíněných problémů.

Jednou z nejčastěji zmiňovaných možných budoucích platforem jsou celulární automaty (CA). CA je masivně paralelní výpočetní systém založený na lokální interakci jednoduchých buněk, které mohou nabývat různých stavů, periodicky se měnících na základě stavů svých sousedů. Touto problematikou bych se chtěl zabývat ve své disertační práci na téma Sebeopravující se masivně paralelní výpočetní architektury.

Cílem tohoto článku je na CA ukázat různé metody sebereplikace, které souvisejí i s problémy sebeopravy či samokonfigurace. Článek je členěn následovně: Druhá kapitola se věnuje stručnému přehledu problematiky CA, v další kapitole jsou podrobněji probrány některé významné techniky sebereplikace a čtvrtá kapitola naznačuje další možný postup výzkumu. Vše je potom shrnuto v závěru.

2 Celulární automaty

Jak už bylo zmíněno v úvodu, CA je masivně paralelní výpočetní systém, který se skládá z velkého množství jednoduchých buněk, které mohou nabývat předem definovaných stavů. Můžeme se setkat

s jednorozměrnými, dvourozměrnými i třírozměrnými automaty. Nejčastěji jde o 2D ve tvaru čtvercové mřížky.

CA můžeme rozdělit na synchronní a asynchronní. To závisí na způsobu aktualizace stavů buněk. V synchronním CA dochází periodicky u každé buňky ke kontrole stavu okolních buněk. Těmto buňkám se říká okolí a podle jejich tvaru a typu automatu může být prakticky libovolně velké, obvykle jde o čtvercové buňky s 5- nebo 9-okolím (středová buňka se rovněž počítá).

Při každém synchronizačním taktu se zjistí stav okolí každé buňky a zjištěná kombinace se vyhledá v seznamu pravidel. Pravidla jsou zapsána ve tvaru NWCES => C (tedy sever, západ, střed, východ, jih => střed – nadále pracujeme s 2D automatem a 5-okolím). Pokud je pravidlo nalezeno, nový stav na pravé straně pravidla se zapíše do kopie stavů celého CA a po zjištění změn u všech ostatních buněk se stav z této kopie aktualizuje. Pokud pro zjištěný stav okolí žádné pravidlo neexistuje, ponechá se buňka v dosavadním stavu.

Použijeme-li pro všechny buňky stejnou sadu pravidel, říkáme takovému CA uniformní. Má-li však každá buňka svoji vlastní sadu, která se může lišit od ostatních buněk, získáme neuniformní CA. Neuniformní CA je důležitým krokem pro výpočetní univerzalitu, což je vlastnost, kterou bychom chtěli mít u každého výpočetního systému. Byly vytvořeny univerzální uniformní CA s mnoha stavy a složitým sousedstvím (přehled v [2]), nicméně např. u případu 5-okolí a dvou stavů bylo dokázáno, že univerzální CA sestavit nelze [8].

Neuniformita nám umožňuje to, abychom některé buňky použili jako vodiče a další buňky jako logická hradla. Stačí nám vytvořit hradla XOR kvůli křížení vodičů a hradla NAND kvůli kompletní množině logických funkcí, která nám umožní implementovat jakýkoli logický výpočet, čímž získáváme výpočetní univerzalitu. Podrobnější vysvětlení a důkaz v [9].

Pomineme-li problém počáteční konfigurace (tj. nastavení stavů či distribuce vhodných pravidel v neuniformním automatu), můžeme si všimnout toho, že CA funguje zcela autonomně a jen na základě lokálních interakcí a synchronizace. To je důležitá vlastnost právě kvůli urychlení výpočtu. Není třeba složitých algoritmů na zasílání zpráv mezi buňkami/procesory.

Další důležitou vlastností je emergence. To je vlastnost, která je založena právě na lokálních interakcích mezi buňkami CA. (Pozn.: Emergence však není výsadní vlastností CA.) Z jedné buňky a sady pravidel těžko určíme, jak bude vypadat výsledné chování CA s mnoha buňkami. Vhodným návrhem pravidel a také vhodným nastavením počátečního stavu automatu jsme schopni získat velice komplexní chování. Příkladem může být i experiment s jednoduchým automatem zvaný hra Life či princip sebereplikace, který bude vysvětlen níže.

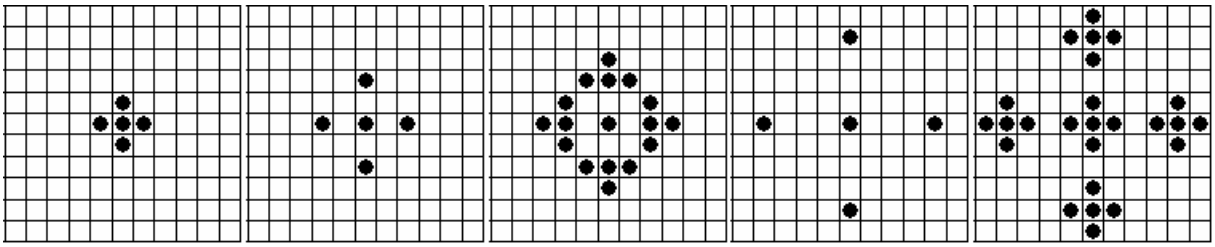
Vhodnou poznámkou na závěr kapitoly je i to, že některé postupy aplikované na CA můžeme použít i na složitější zařízení, protože buňky CA mohou být brány jako abstrakce, např. jednoduchých procesorů. Zároveň existují i některé složitější modely založené na CA, vhodným příkladem je systém Cell Matrix, který byl již i fyzicky implementován [1].

3 Sebereplikace v CA

Sebereplikace je technika, která umožňuje kopírovat struktury vytvořené v CA. Kromě toho, že to je vhodný způsob pro nastavení počáteční konfigurace, může být i prostředkem k znovuobnovení chování po poruše, jak bude později vysvětleno. Zde nejprve zběžně zmíníme dva méně využitelné postupy a poté přejdeme k hlavnímu tématu tohoto článku a sice k replikujícím se smyčkám a jejich vylepšením.

3.1 Sebereplikace pomocí pravidel

Nejjednodušším způsobem k dosažení sebereplikace je vhodná konstrukce přepisovacích pravidel CA. Takových pravidel existuje např. na 2D automatech s dvěma stavy celá řada – za všechny lze uvést pravidlo liché parity, které při dostatečném prostoru dokáže replikovat jakékoli vzory při zdvojnásobení počtu kroků [10]. Průběh takové replikace je ilustrován na obrázku 1.



Obrázek 1: Průběh replikace pomocí pravidla liché parity na 2-stavovém CA 11x11 buněk v pěti krocích

Bohužel, tento přístup, pokud by byl použit k obecné sebereplikaci, je příliš naivní a velmi těžko lze najít nějaké jeho využití, protože pozice replikovaných vzorů se neustále mění a automat je omezen jen na dva stavy.

3.2 Von Neumannův konstruktor

Opustíme-li ideu jednoduchých pravidel a podíváme se na sebereplikaci z jiné strany, dříve nebo později narazíme na Von Neumannův konstruktor (VNK), což je replikační stroj, jenž se zrodil nedlouho po samotných CA koncem šedesátých let. Jeho cílem bylo napomoci studiu logické organizace sebereplikujících se struktur [6].

Jde o rozsáhlý CA skládají se s desítek tisíc buněk, které mají 5-sousedství a 29 možných stavů. Tento automat vlastně simuluje Turingův stroj, který používá konstrukční rameno, aby krok za krokem vytvořil kopii sama sebe podle instrukcí na pásce. Po provedení replikace může stroj provádět další užitečné výpočty [7].

Už z popisu vyplývá, že ani tento postup není prakticky využitelný, protože desítky tisíc buněk a více než 20 milionů možných pravidel znamenají obrovské výpočetní nároky. VNK však přinesl několik nových poznatků o použitelnosti některých struktur v CA a přispěl k tomu, aby se začaly zkoumat minimální prostředky nutné pro sebereplikaci [6].

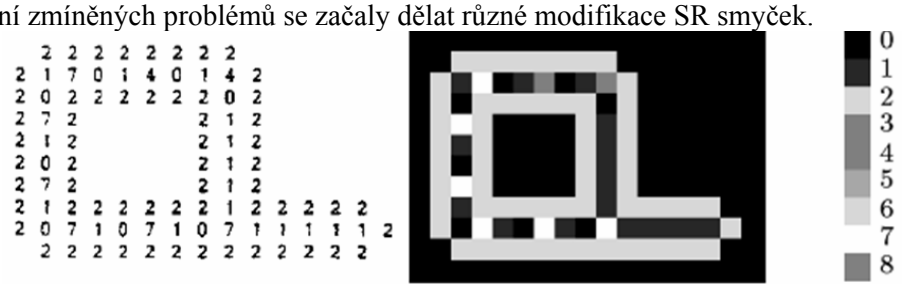
3.3 Sebereplikující se smyčky

Tento nejnovější přístup k sebereplikaci byl objeven v roce 1984 Langtonem [3]. Hlavní myšlenkou je zakódování informace řídicí reprodukci do struktury „organismu“. Už není požadována univerzalita a CA má 8 stavů a zhruba 280 pravidel (+ další symetrická pro ostatní rotace). Smyčka je struktura vytvořená ze stavů CA. Počáteční konfigurace smyčky je ilustrována na obrázku 2. V dnešní době existuje i několik dalších základních typů smyček, které se liší velikostí, nepřítomností zapouzdření či počtem stavů. Nejde však o zásadní změny, proto se nadále budeme zabývat Langtonovou smyčkou (běžně uváděnou jako SR – self-replicating).

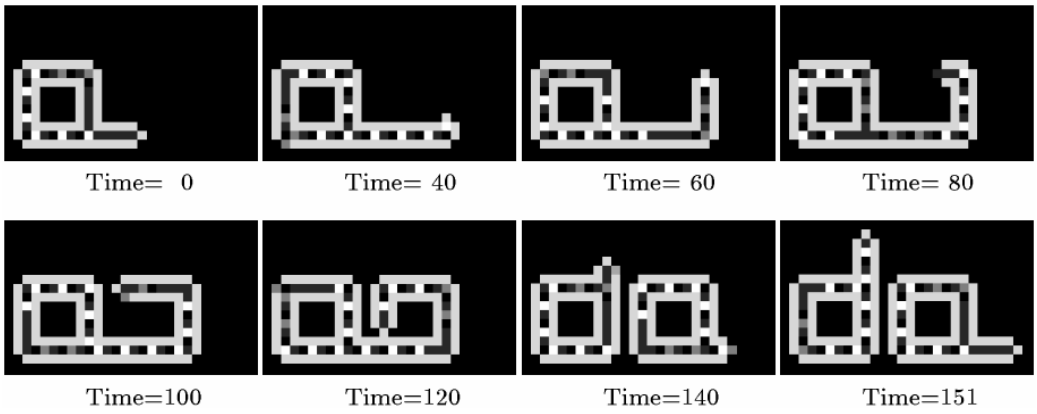
Sebereplikace v tomto případě funguje tak, že vhodnou kombinací pravidel a počáteční konfigurace docílíme specifické chování: Smyčka na základě informace obíhající uvnitř pouzdra složeného ze stavů 2 vytváří na konci „pupeční šňůry“ své další pokračování, které při posuvu informace pouzdrem zahýbá podle toho, který stav je právě na konci. Po 151 synchronizačních krocích CA dojde ke stavu, který je ilustrován na obrázku 3. „Pupeční šňůra“ se v závěru replikace sama rozloží a na nejbližší volné straně smyčky „vypučí“ nové volné zakončení, které vytváří další kopii smyčky v jiném směru.

Tato implementace SR smyčky, má však několik nedostatků. Prvním problémem je, že v případě nedostatku místa se funkce smyčky zasekne a postupně se tak v prostoru CA vytváří pole tzv. „mrtvých“ smyček, které se již dále nemohou replikovat. Druhým problémem je, že když dojde k nedefinovanému stavu nebo když smyčka narazí na chybu v prostoru CA (náhodně změněný osamocený stav), funkce smyčky se buď pokazí nebo se rovněž zasekne. Dalším neméně důležitým problémem je skutečnost, že smyčka se jen replikuje a nevykonává žádnou jinou užitečnou funkci.

K vyřešení zmíněných problémů se začaly dělat různé modifikace SR smyček.



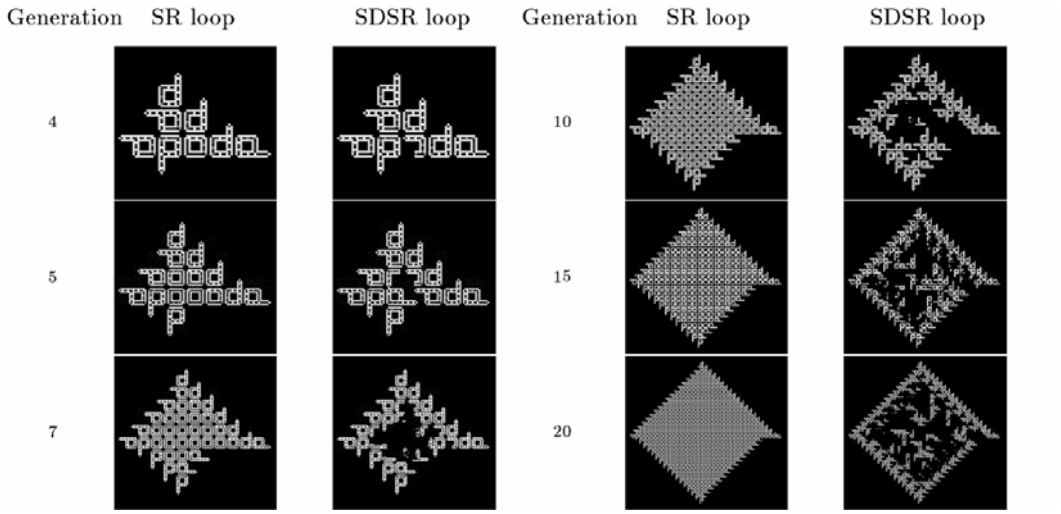
Obrázek 2: Počáteční konfigurace SR smyčky v číselné a barevné reprezentaci stavů [8][7]



Obrázek 3: Cyklus replikace SR smyčky během 151 kroků [7]

3.4 Rozpouštějící se smyčky

Autorem rozpouštějící se smyčky (SDSR – self-dissolving) je Hiroki Sayama [7]. Jde o modifikaci SR smyčky, která má přidaný jeden stav, přičemž při přechodu do něj se začne struktura smyčky rozkládat a měnit se v „klidné“ stavy (stav 0 na obrázku 2). Dosáhneme tím chování, kdy smyčka při svém „úmrtí“ či chybě rozloží své tělo a uvolní tak prostor dalším replikantům.



Obrázek 4: Porovnání života kolonie SR a SDSR smyček od 4. do 20. generace [7]

Z pohledu sebeopravy jde o důležitou vlastnost, protože nejen že je smyčka vhodnou konstrukcí pravidel schopna detekovat chyby v prostoru CA a při replikaci, ale je zároveň schopna odstraněním špatné kopie umožnit růst nového potenciálně správného organismu. Na obrázku 4 je ukázka porovnání života kolonie SR a SDSR smyček.

Z pohledu funkčnosti tím ale vzniká další problém: Namísto toho, aby se replikovaná smyčka zastavila a prováděla nějaký užitečný výpočet, dochází poměrně často k jejímu rozložení a nahrazení další smyčkou. Pokud by výpočet trval méně než je životní cyklus smyčky, neznamenalo by to problém, avšak v opačném případě by se tomuto nežádoucímu chování muselo zabránit vhodnou modifikací pravidel.

3.5 SR smyčka schopná univerzálních výpočtů

Dalším problémem zmíněným v podkapitole 3.3 je nedostatek funkčnosti u SR smyček. Na tento problém se pokusila najít odpověď skupina kolem Perriera [5], která přidala k SR smyčce Wangův automat (ekvivalentní Turingově stroji). Ke spodní části smyčky byly připojeny pásky s instrukcemi a s daty (omezená délka pásky), které začnou pracovat po dokončení replikace.

To bohužel způsobilo další problém. Aby mohly být ke smyčce připojeny pásky, musela být reprodukce smyčky omezena jen do jednoho směru. Toto omezení se dále pokoušel obejít Tempesti [9], který informace pro vykonávání programu vložil do obvodu smyčky. Tím se však připravil o možnost univerzality, protože podle složitosti výpočtu musí být upravena velikost smyčky.

4 Možné vylepšení metod sebereplikace

V této kapitole bych se chtěl věnovat možnostem dalšího výzkumu v oblasti sebereplikace s ohledem na téma mé disertační práce, které zní „Sebeopravující se masivně paralelní výpočetní architektury“.

Předchozí kapitola ukázala, že existuje množství modifikací sebereplikujících se smyček v CA (mimo to byly některé varianty z důvodů místa vynechány – např. Evoloop [7]). Některé z těchto principů byly již implementovány do reálných platforem na bázi FPGA, přičemž byly zkombinovány s dalšími biologií inspirovanými technikami (např. Cell Matrix, Embryonics).

Cílem mého dalšího snažení by mělo být nalezení slabého místa mezi těmito modely a platformami a vhodné skloubení výše popsaných principů do nového modelu s lepšími vlastnostmi či o akceleraci stávajícího.

Jednou z méně prozkoumaných možností může být tzv. „teorie poziční informace“, což je další z biologií inspirovaných postupů souvisejících s developmentem. V přírodě je chování každé buňky v embryu ovlivněno její pozicí, což např. zaručuje, že mlokovi rostou prsty na koncích končetin a nikoli na hlavě. Podle teorie poziční informace existuje hypotetická látka zvaná morfogen, která rozdíly ve své koncentraci ovlivňuje vznik různých buněčných struktur. Vzniká tak jakási gradientní mapa, podle které se specializují jednotlivé buňky organismu [11]. Chování systému tak neovlivňují pouze pravidla buněk, ale i externí informace, kterou „dodává“ prostředí.

V informačních technologiích byl tento princip vyzkoušen na CA na tzv. problému „francouzské vlajky“ (french flag) [4], kdy buňky mohou nabývat jedné ze tří barev na základě své pozice. Možností dalšího výzkumu je prozkoumat eventuality, které tato teorie skýtá pro konfiguraci výpočetního systému prováděnou sebereplikací, přičemž se nabízí i využití gradientu k rozdělení na různé typy výpočtů v jednom CA.

5 Závěr

V článku byly představeny celulární automaty jako výzkumná platforma pro masivně paralelní výpočetní architektury a byla zdůrazněna role sebereplikace v těchto systémech. Bylo představeno několik technik sebereplikace včetně jejich významných výhod a nevýhod. V poslední kapitole byl

navržen další postup výzkumu, který si klade za cíl vytvořit model masivně paralelní výpočetní architektury založené na CA a na pokročilých metodách sebereplikace. Dále byl představen princip poziční informace, který by mohl být jedním z prostředků na vylepšení či akceleraci takového modelu. S využitím externí informace by mohlo být např. možné provádět replikaci v menším počtu kroků.

6 Dosavadní činnost na disertaci

Největší část činnosti na disertaci dosud zabralo studium literatury. Nastudoval jsem problematiku CA a replikace a rovněž jsem přečetl řadu článků o nanotechnologiích, zejména v oblastech samoorganizace a odolnosti proti poruchám.

Dále jsem za účelem experimentování s CA a sebereplikací vytvořil simulační software.

Kromě toho jsem se zabýval dalšími problémy inspirovanými biologií, a sice evolučním návrhem číslicových obvodů na úrovni tranzistorů [13] a koevolucí řadicích sítí [12].

Literatura

[1] Cell Matrix Corporation home page, http://www.cellmatrix.com, červen 2008

[2] Culik II, K., Hurd, L. P., and Yu, S.: Computation theoretic aspects of cellular automata, Physica D, vol. 45, p. 357-378, 1990

[3] Langton, C.G.: Self-Reproduction in Cellular Automata, Physica D: Nonlinear Phenomena 10(1-2): p. 135-144, 1984

[4] Miller, J., Banzhaf, W.: Evolving the Program for a Cell: From French Flags to Boolean Circuits, in On Growth, Form and Computers (edited by Kumar, S., Bentley, P.J.), Elsevier Academic Press, London, 2003

[5] Perrier, J.Y., Sipper, M., Zahnd.J.: Toward a Viable, Self-Reproducing Universal Computer, Physica D, vol. 97, p. 335-352, 1996

[6] Reggia, J.A., Chou, H.-H., Lohn, J.D.: Cellular Automata Models of Self-Replicating Systems, Advances in Computers, vol. 47, Academic Press, New York, 1998

[7] Sayama, H.: Constructing Evolutionary Systems on a Simple Deterministic Cellular Automata Space, Ph.D. Dissertation, Department of Information Science, Graduate School of Science, University of Tokyo, December 1998

[8] Sipper, M.: Evolution of Parallel Cellular Machines - The Cellular Programming Approach, Lecture notes in Computer Science, vol. 1194, Springer, 1997

[9] Tempesti, G.: A New Self-Reproducing Cellular Automaton Capable of Construction and Computation, Advances in Artificial Life, Lecture notes in Computer Science, vol. 929, Springer, 1995

[10] Wolfram, S.: A New Kind of Science, p. 824, Wolfram Media Inc., Champaign, IL, 2002

[11] Wolpert, L., Jessel, T., Lawrence, P., Meyerowitz, E., Robertson, E., Smith, J.: Pronciples of Development, Third Edition, Oxford University Press, Oxford, 2007

Publikace autora

[12] Žaloudek, L.: Coevolution of Sorting Networks and Training Vectors, Proceedings of the 13th Conference Student EEICT 2007, vol. 1, p. 184-186, FEKT a FIT VUT v Brně, Brno, 2007

[13] Žaloudek, L., Sekanina, L.: Transistor-level Evolution of Digital Circuits Using a Special Circuit Simulator, Proceedings of the 8th International Conference on Evolvable Systems: From Biology to Hardware, Lecture Notes in Computer Science 5216, Springer, 2008 – v tisku