UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Iztok Jeras

Predslike 2D celičnih avtomatov

MAGISTERSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: prof. dr. Branko Šter

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.
Slike so izdelane s pomočjo jezika PGF/TikZ.
Grafi so narisani s pomočjo programa ***gnuplot***.

Namesto te strani **vstavite** original izdane teme diplomskega dela s podpisom mentorja in dekana ter žigom fakultete, ki ga diplomant dvigne v študentskem referatu, preden odda izdelek v vezavo!

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Iztok Jeras,

z vpisno številko 63030393,

sem avtor diplomskega dela z naslovom:

Predslike 2D celičnih avtomatov

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom
  prof. dr. Branko Šter
  in somentorstvom
  prof. [doc.] dr. Ime Priimek
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne xx.xx.2016 Podpis avtorja/-ice:

Zahvala

Na tem mestu se diplomant zahvali vsem, ki so kakorkoli pripomogli k uspešni izvedbi diplomskega dela.

Morebitno posvetilo

Kazalo

Seznam uporabljenih kratic in simbolov

1D .... eno dimenzionalen

2D .... dvo dimenzionalen

CA .... celični avtomat

GoL .... Game of Life (igra življenja)

GoE .... Garden of Eden (stanje brez predslik)

trid .... okolica CA sestevljena iz treh celic

quad .... okolica CA sestevljena iz štirih celic $M_x = M_y = 2$

Povzetek

Medtem ko je računanje predslik 1D celičnih avtomatov dobro raziskan in podrobno dokumentiran problem, je to področje pri 2D celičnih avtomatih manj raziskano. To magistrsko delo poizkuša aplicirati metode razvite za 1D avtomate na 2D problem. Prikazan je algoritem, ki omogočajo štetje in izpis predslik. Razvit je bil s pomočjo grafičnega modela, ki omogoča uporabo teorije grafov pri izračunih in dokazih.

Ključne besede:

celični avtomati, predslike, procesna zahtevnost, reverzibilnost, trid, quad

Abstract

While computing preimages of 1D cellular automata is a well researched and documented problem, for 2D cellular automata there is less research available. This masters thesis attempts to apply methotds developed for 1D automata to the 2D problem. An algorithm is shown, which can count and list preimages. It was developed with the help of a graphical representation, which enables using graph theory for computation and proofs.

Key words:

cellular automata, preimages, ataviser, computational complexity, reversibility, Garden of Eden, Game of Life, trid, quad

Poglavje 1

Uvod

1.1 Celični avtomat kakor model vesolja

Ker lahko vsak univerzalen sistem modelira vsak drugi univerzalen sistem, predpostavimo, da lahko z univerzalnimi CA modeliramo vesolje. Samo modeliranje vesolja je še izven našega dosega, poizkuša pa se vsaj doseči povezave z teoretično fiziko. S stališča informacijske teorije in termodinamike je predvsem zanimiv model gravitacije kakor entropijske sile (Entropic gravity [9]), ki predpostavlja, da je 3D vesolje projekcija procesov, ki se odvijajo na 2D ploskvi. Podobno CA omogočajo opazovanje abstraktnega kopiranja informacij (replikacija) in evolucijo [8].

1.2 Informacijska dinamika

Informacijsko dinamiko CA se najpogosteje opisuje samo kakor reverzibilno ali ireverzibilno, obstaja tudi nekaj člankov, ki opazujejo entropijo sistema. Pogosto je tudi opazovanje dinamike delcev pri Game of Life ali elementarnem pravilu 110. Ne obstaja pa še splošna teorija dinamike informacij v CA. V svojem članku [5] in prispevkih na konferencah [3, 4], sem grafično upodobil predslike trenutnega stanja za 1D problem. Iz upodobitve je videti, da se ponekod izgubi več informacije kakor drugod, kar kaže na možnost izpeljave kvalitativne in kvantitativne teorije dinamike informacij; žal se ta možnost še ni udejanila. Podobno je možno grafično upodobiti predslike 2D CA, ter iz grafov sklepati o izgubi informacij v 2D CA.

1.3 Problem predslik 2D CA

Najbolje teoretično raziskan 2D CA je GoL (Game of Life ali slovensko igra življenja). Ogromno truda je bilo vloženega v raziskovanje delcev in njihove dinamike. S pomočjo osnovnih gradnikov, je mogoče skonstruirati kompleksnejše sisteme, med katerimi so najzanimivejši turingov stroj /cite in univerzalni konstruktor /cite.

Delci so dejansko atraktorji v razvoju CA na končni običajno periodični mreži (thorus). Algoritmi za iskanje predslik so uporabljeni za določitev atraktorjevega korita /cite. V ireverzibilnem celičnem avtomatu se pojavljajo stanja brez predslik imenovana GoE (Garden of Eden ali rajski vrt). Pomensko so GoE nasprotje delcev, saj se nahajajo kar najdlje od atraktorja na robu korita. GoE stanja prav tako kakor delci privlačijo raziskovalce /cite, čeprav v manjši meri kakor delci.

Največ raziskav s področja predslik GoL je bilo opravljenih ravno s ciljem iskanja GoE stanj /cite. S stališča algoritma za štetje predslik je GoE stanje tako, ki ima 0 predslik. Algoritem za štetje predslik je možno pretvoriti v algoritem za preverjanje ali je stanje GoE, tako da se operacije nad celimi števili pretvori v logične operacije nad Boolovimi staji.

Raziskave algoritmov sem se lotil s predpostavko, da je možno opraviti štetje s zahtevnostjo, ki je linearno odvisna od velikosti problema $O(N)$ (n je število opazovanih celic). Čeprav je to možno pri 1D problemu, se izkaže da 2D problem ni tako preprost. Predstavljeni so primeri iz katerih je razvidno, da algoritem z linearno zahtevnostjo ne more pravilno opisati vseh situacij. Zahtevnost opisanega algoritma sicer raste s kvadratom eksponenta eksponentno z eno od dimenzij CA mreže $O(2^2N)$, ja pa možno da obstaja algoritem z nižjo kompleksnostjo.

Opisan algoritem je primerjan z ostalimi doslej znanimi algoritmi. Čeprav s stališča procesne zahtevnosti ne prinaša želenega napredka, pa to da temelji na pregledni grafični upodobitvi daje upanje, da bojo razne optimizacije razvidne bodočim raziskovalcem.

Paulina Léon in Genaro Martínez [7] poizkušata aplicirati De Bruijn-ove diagrame na 2D celične avtomate, doslej je bilo to orodje uporabljeno le na 1D problemih. Točneje, opazujeta dva celična avtomata:'Game of Life' in 'Diffusion rule', s poudarkom na opazovanju stabilnih delcev. De Bruijn-ovi grafi so tudi osnova mojih raziskav, so pa drugače grafično upodobljeni, tako da se jih lahko poveže v opis celotnega celičnega sistema, in niso omejeni na opis predslik ene same celice.

Razni avtorji [2] iščejo vzorce tipa 'Garden of Eden' v celičnem avtomatu

'Game of Life'. Zanimiv je pristop s teorijo končnih avtomatov in regularnih jezikov, ki je v osnovi namenjen eno dimenzionalnim sistemom. Jean Hardouin-Duparc ga je razširil tako, da je celice iz vrstice 2D polja združil v simbole regularnega jezika, zaporedje več vrstic pa predstavlja besedo. Originalni članek je v francoščini, zato še iščem članek, kjer bi pristop opisal v angleščini. Podoben pristop s končnimi avtomati nameravam uporabiti tudi sam.

Doslej sem že razvil napredne algoritme za izračun predslik 1D sistema. Skozi zgodovino so taki algoritmi napredovali, tako da je padala njihova procesna zahtevnost in opisna/implementacijska zahtevnost.

1. 'brute force' algoritmi $O(c^N)$
2. improvizirani algoritmi
3. zasnove matematičnega modela
4. optimalni algoritmi $O(N \log(N))$ ali celo $O(N)$

Iskanje slik 2D sistema je trenutno nekje med improvizacijo in matematičnim modelom. Z magistrsko nalogo bi rad razvil algoritme, ki se nagibajo k optimalnosti.

Magistrsko delo bo obsegalo matematičen model, ki bo predvidoma temeljil na matričnih operacijah, kjer matrike predstavljajo grafe in končne avtomate. Za lažje razumevanje bo problem tudi grafično predstavljen. Algoritem bo implementiran kot računalniški program, ki bo omogočal tudi izris grafične predstavitve problema.

Primerjava s sorodnimi deli bo s stališča procesne zahtevnosti algoritma in glede na to, katere znane probleme bo algoritem sposoben rešiti. Nekaj takih problemov, urejenih glede na zahtevnost, je:

1. določitev, ali obstajajo predslike za dano trenutno stanje sistema
2. štetje predslik
3. naštevanje konfiguracij predslik
4. jezik vseh stanj brez predslik
5. vprašanje reverzibilnosti sistema

Rešitev problema določitve obstoja predslik si že predstavljam. Predvidevam, da bom uspel rešiti še problem preštevanja predslik, in ker je to manjši korak, tudi njihovo naštevanje.

Preostalih problemov se tokrat ne bom loteval. Problem jezika stanj brez predslik bi potreboval teorijo 2D formalnega jezika, ki še ne obstaja. Poleg tega je povezan s problemom reverzibilnosti, ki je na splošno dokazano nerešljiv [6].

Poglavje 2

Definicija 2D celičnega avtomata

Predstavljena definicija 2D avtomata je enostavna in manj formalna v primerjavi z definicijo 1D avtomata v podobnih člankih. Bolj podrobna definicija ni potrebna, saj se opisani problemi za 2D avtomate, še ne povezujejo z drugimi vejami matematike toliko kakor 1D avtomati.

V grafičnih upodobitvi je uporabljena izometrična projekcija, ker je za potrebe analize osnovni 2D mreži dodana tretja dimenzija, ki opisuje prostor predslik.

Osnovni element 2D celičnega avtomata je celica kakor del polja celic. Vsaka celica ima diskretno vrednost c iz nabora stanj celice S. Stanja so običajno kar oštevilčena.

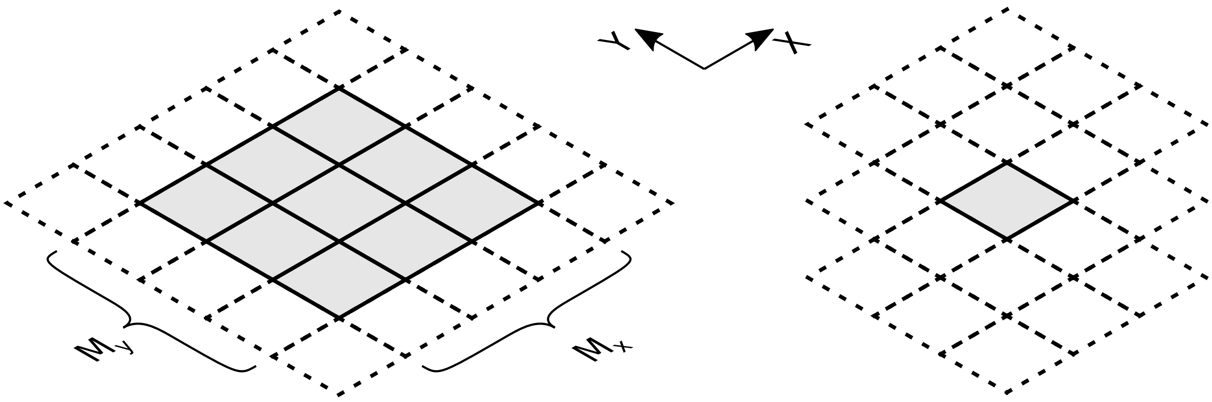
$$c \in S \quad \text{and} \quad S = \{0, 1, 2, |S| - 1\}$$

Mreža polja je lahko pravokotna, šestkotna ali celo kvazikristalna, tukaj se bomo omejili na pravokotno mrežo. Na splošno je velikost polja lahko neskončna, običajno pa se končna polja definira kakor pravokotnik velikosti $N_x \times N_y$ in skupno število celic v končnem polju je $N = |N_x \times N_y|$.

Trenutno stanje neke celice $c_{x,y}$ je odvisno od prejšnjega stanja njene okolice $n_{x,y}$. Podobno se bomo tudi pri obliki okolice omejili na pravokotnik velikosti $M_x \times M_y$. Število celic v okolici je $m = |M_x \times M_y|$. Na voljo je $|S|^m$ možnih okolic.

$$n \in S^m \quad \text{and} \quad S^m = \{0, 1, 2, |S|^m - 1\}$$

Prostorski odnos med okolico in celico v prihodnjem stanju avtomata, ki jo ta okolica določa, ni podrobno definiran. Običajno se smatra, da je celica v sredini okolice, ampak za opisani algoritem to ni nujno pomembno.



Slika 2.1: Okolica celice s podanima dimenzijama $M_x = 3$ in $M_y = 3$.

Preslikava sedanje okolice $n^t_{x,y}$ v prihodnjo istoležno celico $c^{t+1}_{x,y}$ je definiran s tranzicijsko funkcijo f.

$$c^{t+1}_{x,y} = f(n^t_{x,y})$$

Za potrebe iskanja predslik je zanimiva obratna funkcija f^{-1}, ki ob podanem stanju trenutne celice $c^t_{x,y}$ vrne množico okolic, ki se preslikajo v to vrednost.

$$f^{-1}(c^t) = \{n^{t-1} \in S^m \mid f(n^{t-1}) = c^t\}$$

Dodaten pogoj za predsliko polja celic je, da se morajo okolice sosednjih celic ujemati povsod, kjer se prekrivajo.

Podana konstrukcija mreže predslik in algoritem za izračun predslik omogočata uporabo bolj splošne definicije, kjer je za vsako celico definiran lasten nabor predslik, ki je neodvisen od stanja celice. Ta posplošitev je uporabljena za konstrukcijo umetnih mrež predslik, ki poudarjajo konkretne probleme, ki definirajo kompleksnost algoritma.

$$n^{t-1} \in S^m$$

Za dano polje celic velikosti $N_x \times N_y$ in z odprtimi robovi, je možno izračunati prihodnje stanje polja velikosti $(N_x - (M_x - 1)) \times (N_y - (M_y - 1))$. V primeru, če so robovi polja ciklično zaprti (polje v obliki thorusa), je pa polje prihodnjega stanja enako veliko kakor polje sedanjega. Podobno velja za računanje predslik, za sedanje polje velikosti $N_x \times N_y$ je za odprte robove velikost polja predslik $(N_x + (M_x - 1)) \times (N_y + (M_y - 1))$. Za ciklične robove pa sta velikosti enaki.

Poglavje 3

Konstrukcija mreže predslik

Mreža predslik je grafični konstrukt, ki omogoča upodobitev posameznih pojmov, iz definicije celičnih avtomatov, kakor ločene grafične elemente. Odnosi med temi elementi definirajo pravila na katerih se gradijo algoritmi za iskanje predslik.

3.1 De Bruijnov diagram

Osnovani element grafične upodobitve je De Bruijinov diagram. V osnovi ta obravnava ciklične premike končnih zaporedij simbolov, ter njihovo prekrivanje. Vozlišča v diagramu so vsa možna končna zaporedja, povezave med njimi pa definirajo kako se ta zaporedja prekrivajo med seboj.

Pri 1D celičnih avtomatih se problem neposredno preslika na De Bruijinov graf. McIntosh in njegova skupina uporabljajo za analizo te De Bruijinove grafe neposredno. Sam sem pa razvil modificiran graf, kjer so vozlišča podvojena, in gredo poti vedno od originala proti dvojniku. To omogoča veriženje grafov, in razširitev osnovnega De Bruijinovega grafa, ki opisuje okolico ene celice v verižen graf, ki opisuje verigo celic.

??? [] uporabi podoben pristop za analizo 2D problema, tukaj pa je uporabljen nekoliko drugačen pristop. Za potrebe opisa 2D celičnih avtomatov, je bila elementom De Bruijinovega grafa dodana nova dimenzija. Povezave med vozliči se spremenijo v ploskve, in vozlišča se spremenijo v robove ploskev.

Elementi celičnega avtomata, ki se preslikajo v graf so:

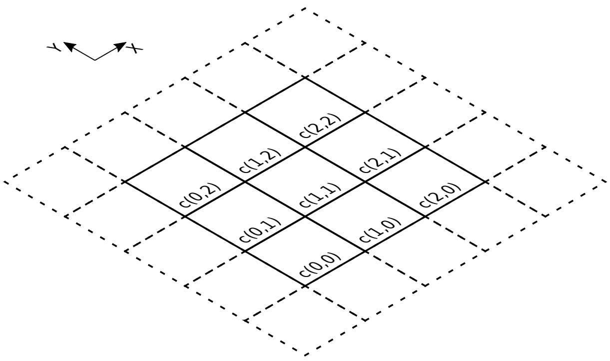
- nabor vseh možnih okolic celice postane nabor vseh ploskev (slika 3.5)
- nabor prekrivanj okolic v smeri 2D dimenzij (slika 3.2) postane nabor povezav

- nabor prekrivanj okolic v diagonalni smeri (slika 3.3) postane nabor vozlišč

Elemente celičnega avtomata, kakor okolice in prekrivanja okolic je potrebno indeksirati, tako da se lahko vsakemu elementu pripiše unikatna števna vrednost. Vsaki okolici je pripisana zaporedna vrednost, ki je konstruirana kakor m mestno število v S-iškem številskem sistemu (za podane primere dvojiški). Cifre si sledijo od spodaj levo do zgoraj desno znotraj okolice.

$$n = \sum |S|^{yN_x+x} \cdot c_{x,y}$$

$$n \in \{0, 1, \ldots N_xN_y - 1\}$$



Slika 3.1: Indeksiranje okolice celice s podanima dimenzijama $M_x = 3$ in $M_y = 3$.

Celice se v smeri dimenzije X prekrivajo za ploskev velikosti $O_x = M_x - 1$ in $O_y = M_y$ (slika 3.2). To ob indeksiranju da nabor:

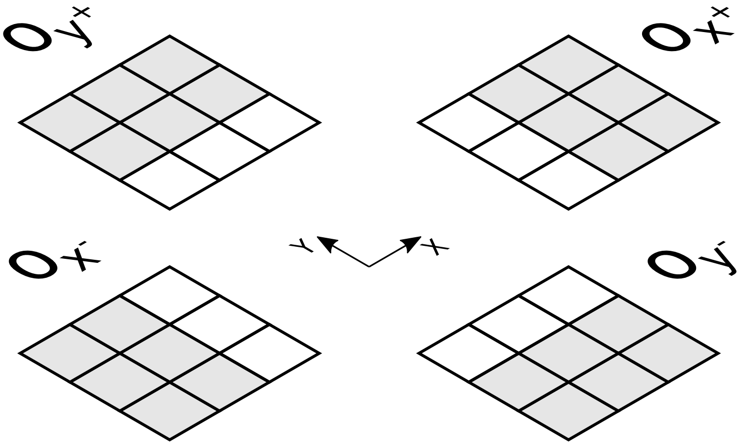
$$o_x = \sum |S|^{y(M_x-1)+x} \cdot c_{x,y}$$

$$o_x \in \{0, 1, \ldots (M_x - 1)N_y - 1\}$$

Celice se v smeri dimenzije Y prekrivajo za ploskev velikosti $O_x = M_x$ in $O_y = M_y - 1$ (slika 3.2). To ob indeksiranju da nabor:

$$o_y = \sum |S|^{yM_x+x} \cdot c_{x,y}$$

$$o_y \in \{0, 1, \ldots M_x(N_y - 1) - 1\}$$

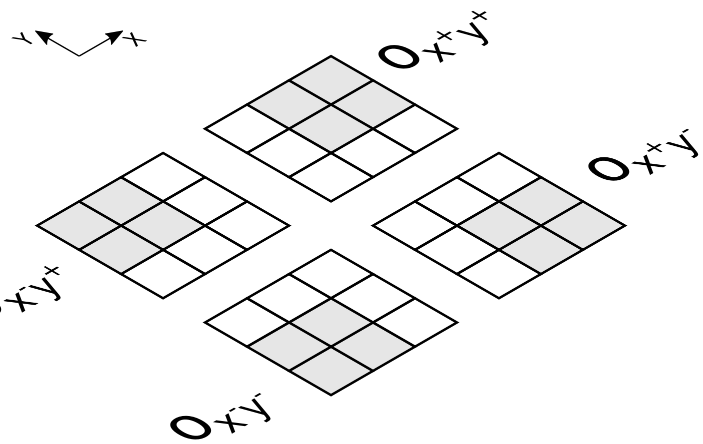


Slika 3.2: Prekrivanje okolic sosednjih celic v smeri dimenzij X in Y, za velikost okolice $M_x = M_y = 3$.

Celice se v diagonalni smeri prekrivajo za ploskev velikosti $O_x = M_x - 1$ in $O_y = M_y - 1$ (slika 3.3). To ob indeksiranju da nabor:

$$o_{xy} = \sum |S|^{y(M_x-1)+x} \cdot c_{x,y}$$

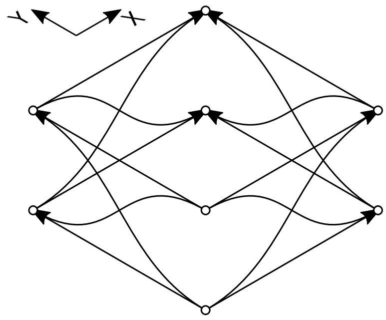
$$o_{xy} \in \{0, 1, \dots (M_x - 1)(N_y - 1) - 1\}$$



Slika 3.3: Prekrivanje okolic sosednjih celic v diagonalni smeri, za velikost okolice $M_x = M_y = 3$.

Nastali graf (slika 3.4) ima poleg vozlišč in povezav med njimi tudi ploskve.

Ploskve bi v teoriji grafov opisali kakor zanke v grafu, z dodano omejitvijo, da mora vsako vozlišče in povezava v zanki pripadati drugemu prekrivanju okolic.



Slika 3.4: Mreža ene celice za binarni CA z okolico quad $M_x = M_y = 2$.

3.2 Mreža

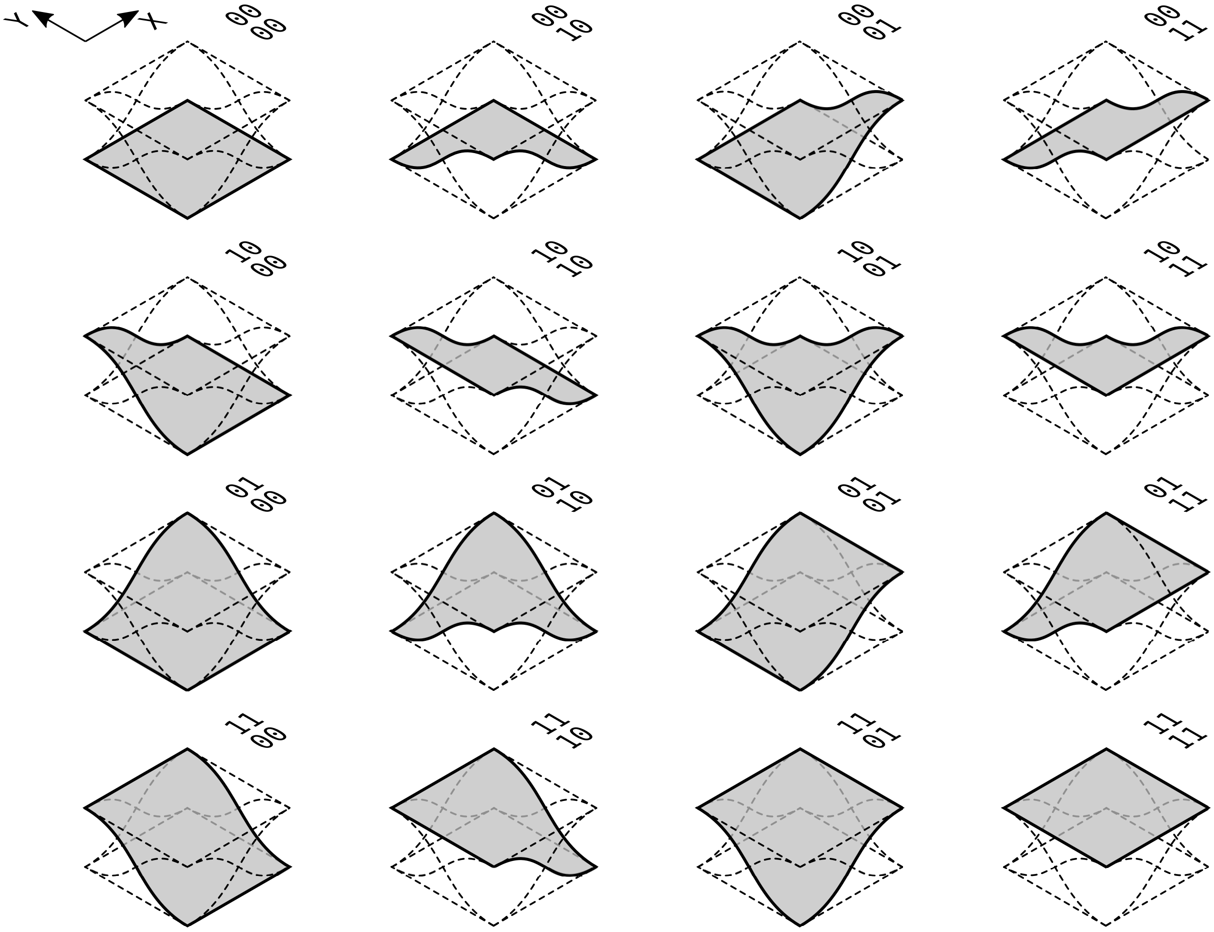
Diagrame, ki opisujejo preteklost posamezne celice, je možno sestaviti v mrežo, ki opisuje polje več celic. Nabor predslik celotnega polja je ekvivalenten naboru vseh zveznih ploskev, ki prekrivajo celotno polje in, ki jih je možno sestaviti iz naborov ploskev diagramov posameznih celic.

Preslikava iz zvezne ploskve v mreži predslik v konfiguracijo polja celic predslike je enolična. Najlažje je razumeti preslikavo za okolico velikosti 3×3, od vsakega odseka ploskve za posamezno celico se vzame centralno celico, na koncu pa se doda še za eno celico širok rob okoli celotne ploskve.

Podani primeri uporabljajo binarni CA z quad okolico. Za ta CA je velikost diagonalnega prekrivanja okolic ena sama celica, Posledično ima nabor vozlišč le dve vrednosti, ki neposredno predstavljajo vrednosti celic v predsliki (slika 3.6).

Razširitev diagrama ene celice v mrežo lahko dokažemo z indukcijo. Dokazati želimo, da je neka konfiguracija celic predslika dane sedanjosti če in samo če je ta ekvivalentna zvezni ploskvi v mreži predslik.

Prvi element: Iz definicije velja, da je za eno samo celico v mreži nabor ploskev enak naboru vseh predslik. **Naslednji element:** Obstoječi mreži predslik za več celic z danim naborom zveznih ploskev dodamo novo celico. Ploskev iz nabora dodane celice se zvezno veže z obstoječim naborom zveznih ploskev natanko v primeru, ko se z eno od ploskev ujema v robu (povezavi med vozliščema). Temu je tako, ker so indeksi vozlišč in povezav ekvivalentni vrednosti prekrivanj okolic.

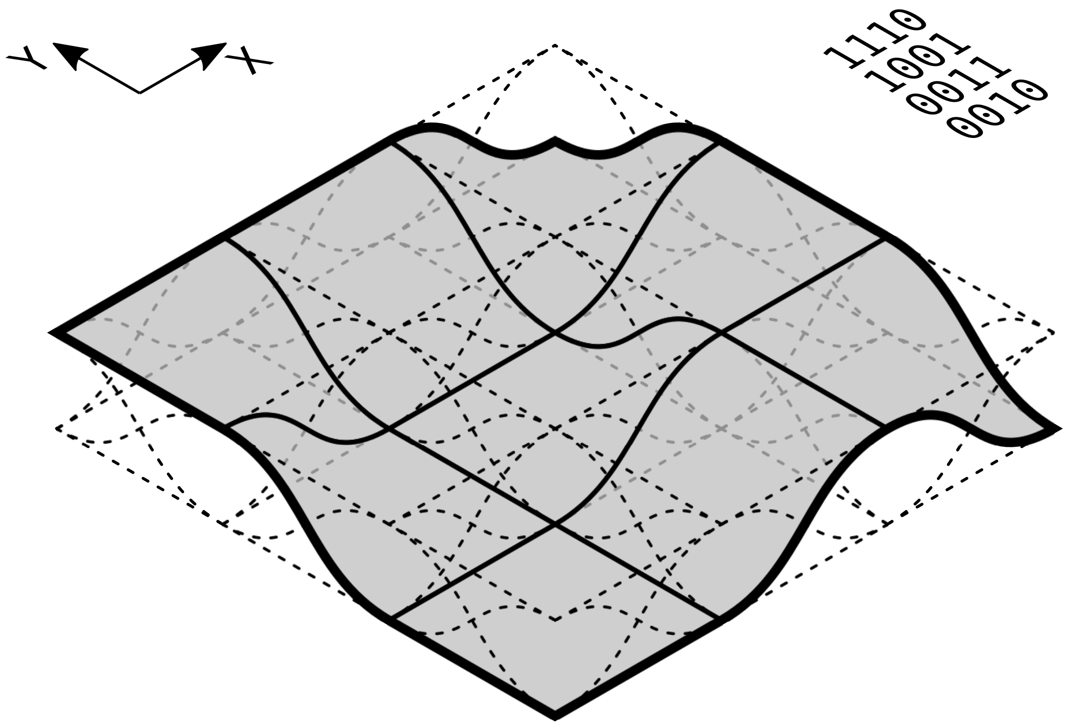


Slika 3.5: Nabor ploskev za vse možne okolice za binarni CA z okolico quad $M_x = M_y = 2$.

3.3 Robni pogoji

Pri 1D CA so robni pogoji definirani na dveh koncih, ki omejujejo končno število celic na neskončni premici. Če je 1D CA definiran kakor daljica je robni pogoj samo eden. Robni pogoj definira katere okolice (vozlišča pri mreži predslik za 1D CA) in s kakšnimi utežmi so na voljo ob robu. Lahko si jih predstavljamo tudi kakor vpliv, ki ga ima neskončen poltrak celic, ki sega izven roba opazovane konfiguracije.

Pri 2D CA je robni pogoj definiran na sklenjeni poti okoli ploskve opazovane konfiguracije celic. V mreži predslik za 2D CA povezave med vozlišči definirajo rob ploskve (slika 3.6). Na splošno ima vsaka ploskev v mreži svoj rob,



Slika 3.6: Mreža velikosti $N_x = 3$ in $N_y = 3$ za binarni CA z okolico quad. Poudarjena je ena zvezna ploskev (konfiguracija pripadajoče predslike je izpisana) in njen rob.

robni pogoj določa kako je ta ploskev obravnavana. Ker uporabna vrednost splošnega robnega pogoja še ni znana, in bi splošnost izrazito povečala zahtevnost algoritma za iskanje predslik, so tukaj vsi robovi obravnavani enako. Temu bomo rekli odprt rob, ker ta ne definira nobenih omejitev, katere zvezne ploskve v mreži predslik so dovoljene in katere ne.

Obstaja še eden enostaven robni pogoj, ki je definiran za ciklično sklenjena končna CA polja. Ta tip robnega pogoja tukaj ne bo obravnavan, ker še dodatno poveča kompleksnost algoritmov.

Poglavje 4

Algoritem za štetje in izpis predslik

Pri 1D CA ima algoritem za štetje predslik linearno $O(N)$ procesno in pomnilniško zahtevnost. Posplošeno to pomeni, da se vsaka celica pojavi v izračunu samo enkrat. Dejansko ima vsak algoritem tudi logaritmično komponento, saj število bitov potrebnih za zapis števcev raste logaritmično s številom celic. Algoritem za izpis predslik ima tudi vedno eksponentno maksimalno kompleksnost, saj maksimalno in povprečno število predslik raste eksponentno v odvisnosti od števila celic.

Za 2D CA se je izkazalo, da obstajajo problemi, ki niso rešljivi z linearno kompleksnostjo. Prikazani algoritem ima maksimalno eksponentno kompleksnost v odvisnosti od posamezne dimenzije $O(S^{N_x}S^{N_y})$. Za sedaj še odprta možnost za obstoj algoritma z kompleksnostjo med linearno in eksponentno zahtevnostjo.

Opisani algoritem za štetje predslik razdeli polje celic na vrstice v dimenziji X. Delitev po stolpcih v dimenziji Y bi bila ekvivalentna, tako da je to arbitrarna odločitev za podani algoritem. Na podlagi zunanjega robnega pogoja, je najprej poiskan nabor predslik za prvo vrstico. Nabor predslik je izračunan kakor uteži za robove na nasprotni strani. Te uteži so uporabljene kakor vhodni robni pogoj za naslednjo vrstico. Robne uteži izračunane za zadnjo vrstico predstavljajo število vseh predslik.

Algoritem za izpis predslik je nadaljevaje algoritma za štetje. Starta se z znanim številom predslik, ki ga je dalo štetje in se izpisuje predslike po vrsticah v obratnem vrstnem redu kakor je potekalo štetje.

4.1 Procesiranje v eni dimenziji

Procesiranje se začne z eno dimenzionalnim nizom celic (vrstico). Vsaki celici v nizu pripada lastna mreža predslik, eno dimenzionalnemu nizu celic posledično pripada povezan niz mrež predslik (slika 4.1 mreža **1**). Vsaka segment mreže v nizu ima svoj nabor veljavnih okolic, ki je definiran s tranzicijsko funkcijo (ali pa je posplošeno poljuben nabor). Na začetku se ploskve okolic sosednjih celic še ne povezujejo v zvezno ploskev čez celoten niz. Izločiti je potrebno vse ploskve okolic, ki se ne povezujejo s okolicami svojih sosednjih celic.

Ker se vsaka vrstica povezuje s predhodno in naslednjo vrstico, je potrebno upoštevati tudi zveznost ploskev na prehodu med vrsticami. Ta povezava med vrsticami je prerez opazovane ploskve na meji med vrsticama. Vsaka ploskev je obravnavana posebej in prerez je izražen kakor robni pogoj. Začetni robni pogoj za vsako vrstico je nabor poti med vozlišči na začetku vrstice. Vsaka pot je obravnavana posebej (slika 4.1 mreža **2**, odebeljena pot).

Začetni robni pogoj se aplicira tako, da se izloči iz obravnave vse ploskve, ki nimajo skupnega roba z robno potjo (slika 4.1 prehod iz mreže **1** v mrežo **2**). Za tem korakom, so vse ploskve definirane v dveh od štirih vozlišč, do nezveznosti lahko prihaja le še na robu nasprotnem začetnemu robu.

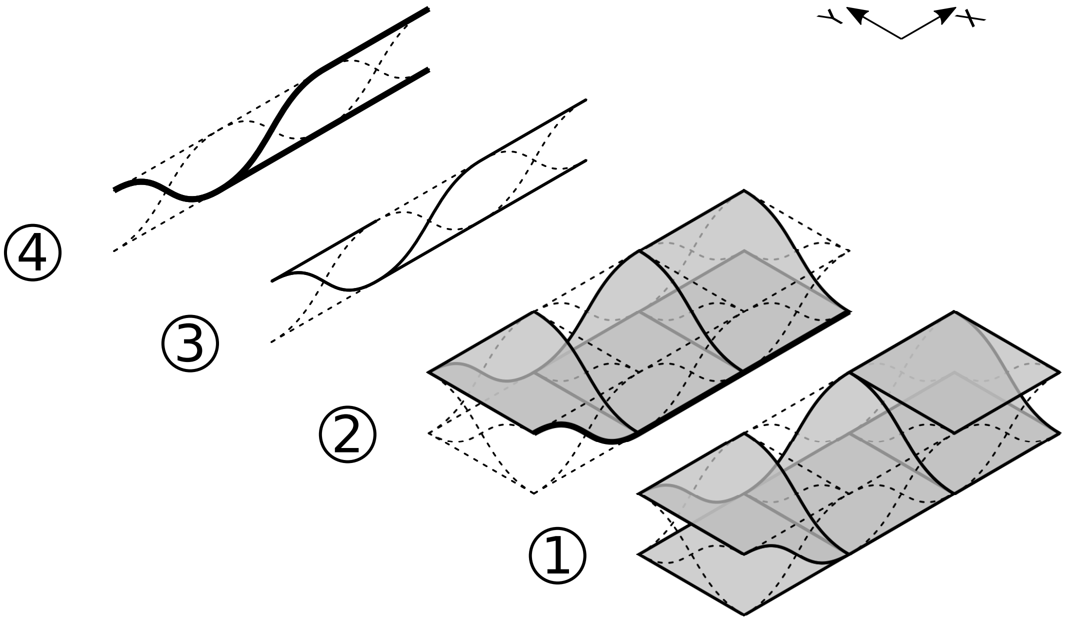
Problem se reducira iz tri dimenzionalne mreže in procesiranja ploskev v dvo dimenzionalno mrežo, kjer se procesirajo poti med vozlišči (slika 4.1 prehod iz mreže **2** v mrežo **3**). To je problem, ekvivalenten iskanju predslik (zvenih poti v grafu) za 1D CA, za kar se uporabi algoritem opisan v []. Rezultat so poti na prerezu med ploskvami (slika 4.1 mreža **4**, končni robni pogoj je odebeljen).

Po procesiranju nabora vseh začetnih robnih pogojev, nastane nabor vseh končnih robnih pogojev. Ta nabor se v naslednjem koraku uporabi kakor začetni robni pogoj za naslednjo vrstico.

Nabor vseh možnih poti na meji vrstice raste eksponentno z dolžino vrstice. Število vseh poti se izračuna iz števila vejitev na vsakem vozlišču in dolžine poti. Za deljenje po vrsticah v dimenziji X je ta nabor $(M_y - 1)^{N_x+1}$. Posledično kompleksnost algoritma raste eksponentno z dolžino vrstice $O(C^{N_x})$.

4.2 Procesiranje v drugi dimenziji

V drugi dimenziji se procesira niz vrstic. Vsaka vrstica se začne in konča z robnim pogojem, ti robni pogoji so prerezi ploskev v celotni mreži predslik. Za tem, ko so procesirane vse vrstice, je znan končni rob nabora vseh zveznih ploskev na celotni mreži predslik. Če je uporabljena boolova algebra je znan



Slika 4.1: Mreža velikosti $N_x = 3$ in $N_y = 3$ za binarni CA z okolico quad. Poudarjena je ena zvezna ploskev (konfiguracija pripadajoče predslike je izpisana) in njen rob.

le obstoj predslik, če pa je uporabljeno množenje in seštevanje, je na koncu znano tudi število predslik.

4.3 Izpis predslik

Ni nujno, da se vsak vrstični začetni pogoj preslika v nabor končnih pogojev. Možno je da nobena pot na končnem robu ne zadošča začetnemu pogoju in mreži predslik. Torej po prejšnjih korakih še ni točno določeno, katere ploskve se združujejo v zvezno celoto in katere ne.

Procesiranje po drugi dimenziji v obratni smeri kakor pri štetju, izloča še preostale slepe poti. Hkrati je možno še izpisovati predslike. Že na začetku procesiranja v obratni smeri, je znano število vseh predslik, kar omogoča rezervacijo pomnilnika. Končne robne poti in njihove uteži, se uporabijo za popisati zadnjo vrstico celic v predslikah. Z vsakim korakom v obratni smeri se popišejo še nove vrstice.

Algoritem tukaj ni podrobno opisan, je pa preprosta razširitev algoritma uporabljenega za 1D CA opisanega v [].

4.4 Nezmožnost procesiranja z linearno kompleksnostjo

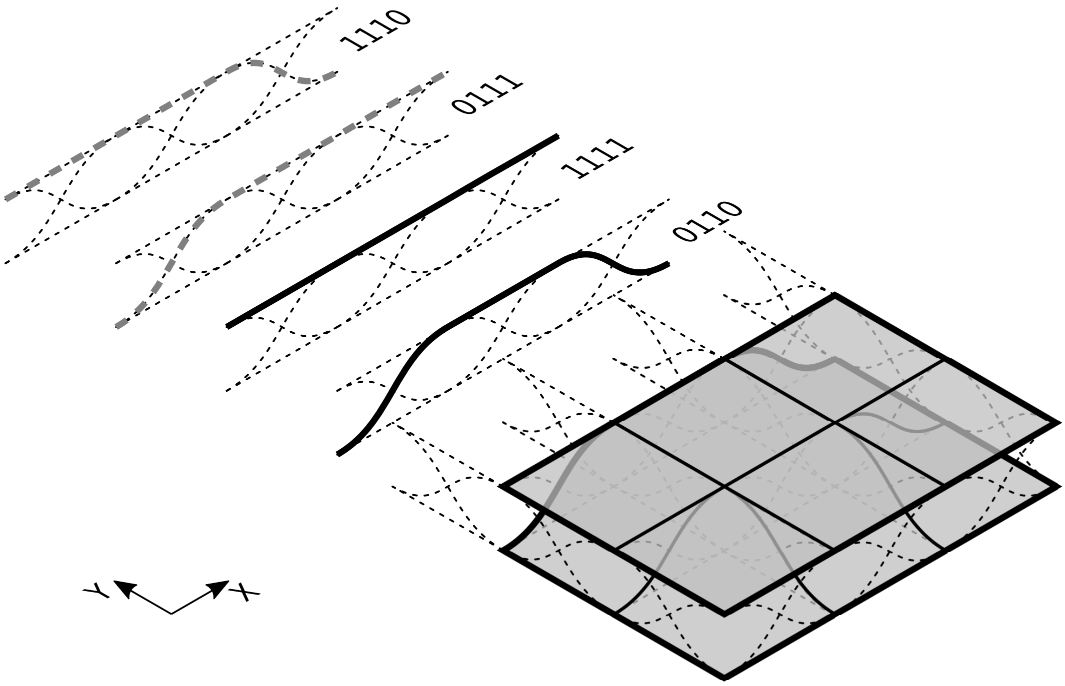
Podan je primer, ki kaže zakaj procesiranje z linearno kompleksnostjo ni mogoče.

Cilj raziskovalnega dela za to raziskovalno nalogo je bil poiskati učinkovit algoritem za iskanje predslik 2D CA. Na podlagi izkušenj z 1D CA sem optimistično pričakoval, da bo možno problem rešiti v linearnem času.

Algoritem v linearnem času, bi vsako celico obravnaval le enkrat ali na splošno bi bilo število obravnav majhna konstanta (4 krat, če se izvaja procesiranje skozi mrežo predslik v 4 smereh/prehodih). Tak algoritem predpostavlja, da je možno celoten nabor začetnih robnih pogojev upoštevati hkrati v enem samem prehodu. Po nekaj poizkusih sem ugotovil da temu ni tako. Nekatere poti iz nabora je potrebno upoštevati ločeno.

V podanem primeru (slika 4.2) se prvi dve vrstici zaključita naborom dveh končnih poti **0110** in **1111**. Če bi te dve poti uporabili hkrati kakor začetni robni pogoj za naslednjo vrstico, bi dobili enak rezultat kakor, če bi bili del robnega pogoja še poti **0111** in **1110**. Iz mreže je razvidno, da te dve poti nista preseka neke zvezne ploskve. Posledično hkratno obravnavanje celotnega nabora robnih pogojev ni možno, kar pomeni, da je število obravnav neke celice odvisno od velikosti problema.

Zgoraj opisani algoritem izrecno obravnava vsak robni pogoj iz nabora ločeno. Je pa videti, da bi bilo možno hkrati obravnavati robove, ki nimajo nobenega skupnega vozlišča. Najbrž obstajajo tudi druge optimizacije algoritma, ki lahko zmanjšajo procesno kompleksnost.



Slika 4.2: Ovira za procesiranje z linearno zahtevnostjo.

Poglavje 5

Primerjava z znanimi algoritmi

Obstoječi algoritmi za predslike 2D CA se osredotočajo izključno na GoL. Največ algoritmov je namenjenih iskanju GoE stanj, skratka preverjajo le obstoj predslik, in jih ne štejejo ali izpisujejo.

5.1 Iskanje GoE

5.2 Iskanje predslik

[1] `https://github.com/PeterBorah/atabot` `https://github.com/bryanduxbury/cellular_chronometer/tree/master/rb` `https://www.kaggle.com/c/conway-s-reverse-g`

5.3 Polni algoritmi

5.3.1 Sklepne ugotovitve

Sklepne ugotovitve naj prikažejo oceno o opravljenem delu in povzamejo težave, na katere je naletel kandidat. Kot rezultat dela lahko navede ideje, ki so nastale med delom, in bi lahko bile predmet novih raziskav.

Slike

Tabele

Literatura

[1] Yossi Elran. Retrolife and The Pawns Neighbors. *The College Mathematics Journal*, 43(2):147–151, 2012. Dostopno na `http://www.jstor.org/stable/10.4169/college.math.j.43.2.147`. 19

[2] Christiaan Hartman, Marijn J. H. Heule, Kees Kwekkeboom, in Alain Noels. Symmetry in Gardens of Eden. *Electronic Journal of Combinatorics*, 20, 2013. 4

[3] Iztok Jeras. Solving cellular automata problems with SAGE/Python. Objavljeno v Andrew Adamatzky, Ramón Alonso-Sanz, Anna T. Lawniczak, Genaro Juárez Martínez, Kenichi Morita, in Thomas Worsch, editors, *Automata 2008: Theory and Applications of Cellular Automata, Bristol, UK, June 12-14, 2008*, strani 417–424. Luniver Press, Frome, UK, 2008. Dostopno na `http://uncomp.uwe.ac.uk/free-books/automata2008reducedsize.pdf`. 3

[4] Iztok Jeras in Andrej Dobnikar. Cellular Automata Preimages: Count and List Algorithm. Objavljeno v Vassil N. Alexandrov, G. Dick van Albada, Peter M. A. Sloot, in Jack Dongarra, editors, *Computational Science - ICCS 2006, 6th International Conference, Reading, UK, May 28-31, 2006, Proceedings, Part III*, del 3993 of *Lecture Notes in Computer Science*, strani 345–352. Springer, 2006. Dostopno na `http://dx.doi.org/10.1007/11758532_47`. 3

[5] Iztok Jeras in Andrej Dobnikar. Algorithms for computing preimages of cellular automata configurations. *Physica D Nonlinear Phenomena*, 233:95–111, September 2007. 3

[6] Jarkko Kari. Reversibility of 2D cellular automata is undecidable. *Physica D: Nonlinear Phenomena*, 45:379–385, 1990. 5

[7] Paulina A. Léon in Genaro J. Martínez. Describing Complex Dynamics in Life-Like Rules with de Bruijn Diagrams on Complex and Chaotic Cellular Automata. *Journal of Cellular Automata*, 11(1):91–112, 2016. 4

[8] Chris Salzberg in Hiroki Sayama. Complex genetic evolution of artificial self-replicators in cellular automata. *Complexity*, 10:33–39, 2004. Dostopno na `http://www3.interscience.wiley.com/journal/109860047/abstract`. 3

[9] Erik P. Verlinde. On the origin of gravity and the laws of Newton. 2010. 3