UNIVERZA V LJUBLJANI
FAKULTETA ZA RAÄŇUNALNIÅăTVO IN INFORMATIKO

Iztok Jeras

Predslike 2D celiÄŊnih avtomatov

MAGISTRSKO DELO
NA UNIVERZITETNEM ÅăTUDIJU

Ljubljana, 2016

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAÄŇUNALNIÅăTVO IN INFORMATIKO

Iztok Jeras

Predslike 2D celiÄŊnih avtomatov

MAGISTRSKO DELO
NA UNIVERZITETNEM ÅăTUDIJU

Mentor: prof. dr. Branko Åăter

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.
Slike in grafi so narisani s pomoÄŊjo programa za vektorske ilustracije Inkscape [4].

[1] http://creativecommons.org/licenses/by-sa/4.0/
[2] http://www.ipi.si/sl/creative-commons-cc/o-uporabi-licence
[3] http://unlicense.org/
[4] https://inkscape.org

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Večna pot 113
1000 Ljubljana, Slovenija
telefon: 01 47 98 100
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Številka: 159-MAG-RI/2016
Datum: 06. 04. 2016

Iztok JERAS, univ. dipl. inž. el.

L j u b l j a n a

Fakulteta za računalništvo in informatiko Univerze v Ljubljani izdaja naslednjo magistrsko nalogo

Naslov naloge: **Predslike 2D celičnih avtomatov**

Preimages of 2D cellular automata

Tematika naloge:

Raziščite problem iskanja predslik 2D celičnih avtomatov s pomočjo De Bruijn-ovih diagramov. Pri tem se opirajte na znane rešitve za 1D problem. Poiščite algoritem za določitev obstoja predslik in algoritem za njihovo preštevanje ter izpis.
Algoritem naj bo matematično formuliran z enačbami in grafično predstavitvijo ter implementiran s programsko opremo. S stališča procesne kompleksnosti primerjajte novo razviti algoritem z obstoječimi algoritmi, ki se uporabljajo za iskanje stanj tipa 'Garden of Eden' v celičnem avtomatu 'Game of Life'.

M e n t o r:

prof dr. Branko Šter

D e k a n:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

IZJAVA O AVTORSTVU
magistrskega dela

Spodaj podpisani **Iztok Jeras**, z vpisno Åątevilko **63030393**, sem avtor magistrskega dela z naslovom:

Predslike 2D celiÄŊnih avtomatov

S svojim podpisom zagotavljam, da:

- sem magistrsko delo izdelal samostojno pod vodstvom mentorja **prof. dr. Branka Åătera** in somentorstvom **prof. dr. Andreja Dobnikarja**,
- so elektronska oblika magistrskega dela, naslova (slov., angl.), povzetka (slov., angl.) ter kljuÄŊne besede (slov., angl.) identiÄŊni s tiskano obliko magistrskega dela
- in soglaÅąam z javno objavo elektronske oblike magistrskega dela v zbirki ÂżDela FRIÂń.

V Ljubljani, dne 30. avgust 2016 | Podpis avtorja:

Zahvala

Zahvaljujem se prof. dr. Andreju Dobnikarju za pomoÄŊ pri pisanju ÄŊlankov o algoritmih za Åątetje in izpis predslik pri enodimenzionalnih celiÄŊnih avtomatih.

Kazalo

Seznam uporabljenih kratic in simbolov

1D .... enodimenzionalen

2D .... dvodimenzionalen

3D .... tridimenzionalen

CA .... cellular automata (celiÄŊni avtomat)

GoL .... Conway's Game of Life (Conwayeva igra Å¿ivljenja)

GoE .... Garden of Eden (rajski vrt)

trid .... okolica CA, sestavljena iz treh celic na heksagonalni mreÅ¿i

quad .... okolica CA, sestavljena iz Åątirih celic na kvadratni mreÅ¿i

C .... poljubna konstanta

S .... nabor stanj celice

$|S|$ .... Åątevilo moÅ¿nih stanj celice

c .... stanje posamezne celice (celoÅątevilska vrednost)

$c_{x,y}$ .... vrednost celice na koordinatah (x, y) znotraj 2D polja celic

c^t .... vrednost celice v sedanjosti

c^{t+1} .... vrednost celice v prihodnosti (en korak)

N_x .... velikost pravokotnega 2D polja celic v dimenziji X

N_y .... velikost pravokotnega 2D polja celic v dimenziji Y

N .... Åątevilo celic v 2D polju

M_x .... velikost pravokotne 2D okolice celice v dimenziji X

M_y .... velikost pravokotne 2D okolice celice v dimenziji Y

m .... Åątevilo celic v 2D okolici

n .... stanje okolice posamezne celice (celoÅątevilska vrednost)

$n_{x,y}$ .... vrednost okolice celice na koordinatah (x, y) znotraj 2D polja celic

n^{t-1} .... vrednost okolice celice v preteklosti (en korak)

n^t .... vrednost okolice celice v sedanjosti

f .... tranzicijska funkcija, ki definira ÄŊasovno evolucijo avtomata

f^{-1} .... obratna tranzicijska funkcija

o_x .... prekrivanje okolic v dimenziji X

o_y .... prekrivanje okolic v dimenziji Y

o_{xy} .... prekrivanje okolic v diagonalni smeri

Povzetek

Medtem ko je raÄŊunanje predslik 1D celiÄŊnih avtomatov dobro raziskan in podrobno dokumentiran problem, je to podroÄŊje pri 2D celiÄŊnih avtomatih manj raziskano. Za 1D problem poznamo algoritme za Åątetje in izpis predslik s procesno zahtevnostjo linearno odvisno od velikosti problema. MoÅ¿no je tudi doloÄŊiti, ali je 1D celiÄŊni avtomat reverzibilen in kakÅąen je regularen jezik vseh stanj brez predslik. Pri 2D problemu sicer poznamo nekaj algoritmov za iskanje predslik, vendar so slabo teoretiÄŊno raziskani. Vemo tudi, da je problem reverzibilnosti 2D celiÄŊnih avtomatov na sploÅąno neodloÄŊljiv.

MreÅ¿a predslik, ki sem jo razvil za potrebe analize predslik 1D celiÄŊnih avtomatov, se je izkazala za praktiÄŊno orodje pri razlagi algoritmov in pri dokazovanju njihove pravilnosti. Tukaj opiÅąem, kako se mreÅ¿o predslik tvori za 2D celiÄŊne avtomate. ÄŇe je bila ta mreÅ¿a za 1D problem navaden graf, je za 2D problem razÅąirjena v tretjo dimenzijo. Predslike so namesto poti v grafu ploskve na mreÅ¿i. Robni pogoj se spremeni iz uteÅ¿i za vozliÅąÄŊa, ki zakljuÄŊujejo pot v 1D problemu, v uteÅ¿i za sklenjeno pot okoli ploskve predslike v 2D problemu.

Pri razvoju algoritma se je izkazalo, da Åątetja predslik 2D celiÄŊnega avtomata ni mogoÄŊe izvesti v linearni odvisnosti od velikosti problema. Zahtevnost podanega algoritma naraÅąÄŊa eksponentno z velikostjo problema v eni dimenziji. Podani algoritem se ne razlikuje dosti od obstojeÄŊih. CeliÄŊni avtomat razdeli na vrstice in iÅąÄŊe predslike za posamezno vrstico od prve do zadnje. VrstiÄŊne rezultate nato zdruÅ¿i v reÅąitev za celoten 2D problem. Podobno kakor pri algoritmu za 1D probleme, je analiza razdeljena v dva prehoda. V prvem prehodu po vrsticah algoritem preÅąteje predslike, v drugem opcijskem prehodu pa predslike izpiÅąe. Drugi prehod poteka v obratni smeri kakor prvi prehod, tudi procesiranje posamezne vrstice poteka iz drugega zornega kota.

Glavni napredek algoritma vidim v uporabi uÄŊinkovitih reÅąitev za 1D problem pri analizi vrstic ter v uporabi progresivnega kodiranja vmesnih rezultatov, kar lahko zmanjÅąa porabo pomnilnika.

KljuÄŊne besede:

celiÄŊni avtomati, predslike, predhodniki, raÄŊunska zahtevnost, reverzibilnost, rajski vrt, Conwayeva igra Å¿ivljenja, 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. For the 1D problem we know algorithms for counting and listing preimages where computational complexity is a linear function of the size of the problem. It is possible to determine whether a 1D cellular automaton is reversible, and what is the Garden of Eden sequence regular language. For the 2D problem we know a few algorithms, but they are poorly theoretically researched. We also know that the reversibility problem is in general undecidable for 2D cellular automata.

The preimage network, first developed for 1D cellular automata, was proved to be a useful tool for explaining algorithms and for constructing proofs. Here I explain how to construct the preimage network for 2D cellular automata. While for the 1D problem this network is a normal graph, for 2D it was extended into the third dimension. Preimages are transformed from paths in the graph in 1D into surfaces on the network in 2D. Edge conditions are transformed from weights for vertices ending a path in the 1D problem into weights for the closed path around a preimage surface in the 2D problem.

While developing the algorithm, it proved impossible to count preimages of 2D cellular automata with processing requirements growing linearly with problem size. Instead, processing requirements grow exponentially with the size in one of the dimensions. The described algorithm does not differ much from the existing ones. The cellular automaton is split into rows, the preimage list is first determined for each row from the first to the last. The row results are then combined into the result for the whole 2D problem. In a similar fashion to the 1D approach, the algorithm splits into two passes. In the first pass preimages are counted, in the second optional pass preimages are listed. The second pass is performed in the opposite direction, while rows are also observed from the opposite side.

I see the main advantage of the described algorithm in using existing solutions for row processing. Solutions proved to be effective in solving the 1D problem. Using progressive encoding of intermediate solutions also enables reducing memory consumption.

Key words:

cellular automata, preimages, predecessors, ataviser, computational complexity, reversibility, Garden of Eden, Conway's Game of Life, trid, quad

Poglavje 1

Uvod

1.1 CeliÄŊni avtomati kakor model vesolja

Ker lahko vsak univerzalen sistem modelira vsak drug univerzalen sistem, lahko predpostavimo, da lahko z univerzalnimi celiÄŊnimi avtomati (CA) modeliramo vesolje. Samo modeliranje vesolja je Åąe izven naÅąega dosega, poizkuÅąa pa se vsaj pribliÅ¿ati teorijo CA in teoretiÄŊno fiziko. S strani informacijske teorije in termodinamike je predvsem zanimiv model gravitacije kakor entropijske sile (Entropic gravity [26]), ki predpostavlja, da je 3D vesolje projekcija procesov, ki se odvijajo na 2D ploskvi. Z druge strani pa CA tudi omogoÄŊajo opazovanje abstraktnega kopiranja informacij (replikacija) in evolucije [24]. Oba sta pomembna informacijska pojava v naÅąem vesolju.

1.2 Informacijska dinamika

Informacijsko dinamiko CA se najpogosteje opisuje samo kakor reverzibilno ali ireverzibilno, obstaja tudi nekaj ÄŊlankov, ki opazujejo entropijo sistema. Pogosto je tudi opazovanje dinamike delcev pri GoL (Game of Life ali slovensko igra Å¿ivljenja) in elementarnem pravilu 110. Ne obstaja pa Åąe sploÅąna teorija dinamike informacij v CA. V svojem ÄŊlanku [16] in prispevkih na konferencah [11, 13, 15] sem grafiÄŊno upodobil predslike trenutnega stanja za 1D problem. Iz upodobitve je videti, da se ponekod izgubi veÄŊ informacije kakor drugod, kar kaÅ¿e na moÅ¿nost izpeljave kvalitativne in kvantitativne teorije dinamike informacij. Na Å¿alost se ta moÅ¿nost Åąe ni udejanila. Podobno je moÅ¿no grafiÄŊno upodobiti predslike 2D CA, ter iz grafov sklepati o izgubi informacije v 2D CA.

1.3 Problem predslik 2D CA

Najbolje teoretiÄŊno raziskan 2D CA je GoL. Ogromno truda je bilo vloÅ¿enega v raziskovanje delcev in njihove dinamike. S pomoÄŊjo osnovnih gradnikov je mogoÄŊe skonstruirati kompleksnejÅąe sisteme, med katerimi so najzanimivejÅąi Turingov stroj [23] in univerzalni konstruktor [6].

Delci so dejansko atraktorji v razvoju CA na konÄŊni periodiÄŊni mreÅ¿i (torus). Pri 1D CA se algoritmi za iskanje predslik uporabljajo za doloÄŊitev atraktorjevega korita [29]. V vsakem CA, ki ni lokalno injektiven, se pojavljajo stanja brez predslik, imenovana GoE (*Garden of Eden* ali rajski vrt) [19, 20]. Pomensko so stanja GoE nasprotje delcev, saj se nahajajo kar najdlje od atraktorja, na robu korita. Stanja GoE tudi privlaÄŊijo raziskovalce, ÄŊeprav v nekoliko manjÅąi meri kakor delci.

NajveÄŊ raziskav s podroÄŊja predslik GoL je bilo opravljenih ravno s ciljem iskanja stanj GoE. S staliÅąÄŊa algoritma za Åątetje predslik je stanje GoE tako stanje, ki nima nobene predslike. Algoritem za Åątetje predslik je moÅ¿no pretvoriti v manj zahteven algoritem za preverjanje ali je dano stanje stanje GoE, tako da se operacije nad celimi Åątevili pretvori v logiÄŊne operacije nad Boolovimi stanji.

1.4 Algoritmi za iskanje predslik

Raziskave algoritmov sem se lotil s predpostavko, da je moÅ¿no opraviti Åątetje z zahtevnostjo, ki je linearno odvisna od velikosti problema $O(N)$ (N je Åątevilo opazovanih celic). ÄŇeprav je to moÅ¿no pri 1D problemu, se izkaÅ¿e, 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 eksponentno z velikostjo ene od dimenzij polja CA $O(C^{N_x})$, je pa moÅ¿no, da obstaja algoritem z niÅ¿jo kompleksnostjo.

Pomembno vlogo pri razvoju opisanega algoritma ima mreÅ¿a predslik. To je grafiÄŊna upodobitev problema, katere cilj je laÅ¿je razumevanje problema in reÅąitve. Osnova za oblikovanje mreÅ¿e predslik so De Bruijnovi diagrami. Paulina LÃľon in Genaro MartÃŋnez [18] poizkuÅąata aplicirati De Bruijnove diagrame na 2D CA. Doslej je bilo to orodje uporabljeno le na 1D problemih. ToÄŊneje, opazujeta dva CA: 'Game of Life' in 'Diffusion rule', s poudarkom na opazovanju stabilnih delcev. Sam uporabljam De Bruijnove diagrame nekoliko drugaÄŊe, in jih tudi drugaÄŊe grafiÄŊno upodabljam. Posamezen De Bruijnov diagram postane mreÅ¿a predslik ene celice. MreÅ¿e posameznih celic se nato povezujejo, tako da na koncu opisujejo celotno polje celic.

Doslej sem Å¿e razvil napredne algoritme za izraÄŊun predslik 1D CA [16]. Skozi zgodovino so taki algoritmi napredovali, tako da je padala njihova raÄŊunska zahtevnost in opisna/implementacijska zahtevnost.

1. 'brute force' algoritmi $O(C^N)$
2. improvizirani algoritmi
3. zasnove matematiÄŊnega modela
4. optimalni algoritmi $O(N)$

Iskanje predslik 2D CA je trenutno nekje med improvizacijo in matematiÄŊnim modelom.

Opisan algoritem na koncu primerjam z ostalimi doslej znanimi algoritmi. ÄŇeprav s staliÅąÄŊa procesne zahtevnosti ne prinaÅąa Å¿elenega napredka, pa to, da temelji na pregledni grafiÄŊni upodobitvi, daje upanje, da bodo razne optimizacije razvidne bodoÄŊim raziskovalcem.

CeliÄŊni avtomat GoL je definiran z Moorovo okolico velikosti 3×3. Tako velika okolica ima z 9 celicami $2^9 = 512$ moÅ¿nih stanj, zaradi ÄŊesar so ilustracije mreÅ¿e predslik velike in nepregledne. SploÅąno je znana manjÅąa von Neumannova okolica velikosti 3×3 v obliki kriÅ¿a iz 5 celic. Z $2^5 = 32$ stanji bi bile ilustracije bolj pregledne, pa obstajajo tudi okolice z manj celicami. Toffoli [25] je leta 2008 spodbudil raziskovalce k iskanju univerzalnosti za CA z okolicama trid (tri celice na heksagonalnem polju) in quad (Åątiri celice na kvadratnem polju). Powley [22] hitro dokaÅ¿e obstoj univerzalnega avtomata z okolico trid, tako da ustvari avtomat, ki lahko s pomoÄŊjo doloÄŊenega zaÄŊetnega stanja simulira poljubni 1D elementarni celiÄŊni avtomat, na primer pravilo 110. Sam sem obe okolici poznal Å¿e prej. Njuna najboljÅąa lastnost je majhen nabor stanj okolice, $2^4 = 16$ za quad in $2^3 = 8$ za trid. Za primere je bil uporabljen quad, saj so diagrami na kvadratni mreÅ¿i bolj enostavni in pregledni kakor diagrami na heksagonalni mreÅ¿i.

Algoritem je implementiran kot raÄŊunalniÅąki program v jeziku C [12]. KnjiÅ¿nica GMP [1] je uporabljena za zapis celih Åątevil, veÄŊjih od 64 bitov. Poleg samega algoritma za Åątetje in izpis predslik sem pripravil tudi orodje za simulacijo binarnega CA z okolico quad [14]. Obe orodji se lahko zaÅ¿ene kar v internetnem brskalniku.

V slikah je uporabljena izometriÄŊna projekcija, saj je za potrebe analize osnovnemu 2D polju dodana tretja dimenzija, ki opisuje prostor predslik (mreÅ¿a predslik).

[1]The GNU Multiple Precision Arithmetic Library `https://gmplib.org/`

Poglavje 2

Definicija 2D CA

Predstavljena definicija 2D CA je enostavna in manj formalna v primerjavi z definicijo 1D CA v podobnih prispevkih. Bolj formalna definicija ni potrebna, saj se opisani problemi pri 2D CA Åąe ne povezujejo z drugimi vejami matematike toliko kakor pri 1D CA.

Osnovni element 2D CA je celica, ki je del polja celic. Vsaka celica ima diskretno vrednost c iz nabora stanj celice S. Stanja so obiÄŊajno kar oÅątevilÄŊena.

$$c \in S \quad \text{in} \quad S = \{0, 1, \ldots, |S| - 1\} \tag{2.1}$$

MreÅ¿a polja je lahko pravokotna, Åąestkotna ali celo kvazikristalna. Tukaj se bomo omejili na pravokotno mreÅ¿o. Na sploÅąno je velikost polja lahko neskonÄŊna, bolj obiÄŊajna pa so konÄŊna polja, definirana kakor pravokotnik velikosti $N_x \times N_y$. Skupno Åątevilo celic v konÄŊnem polju je $N = N_x \cdot N_y$.

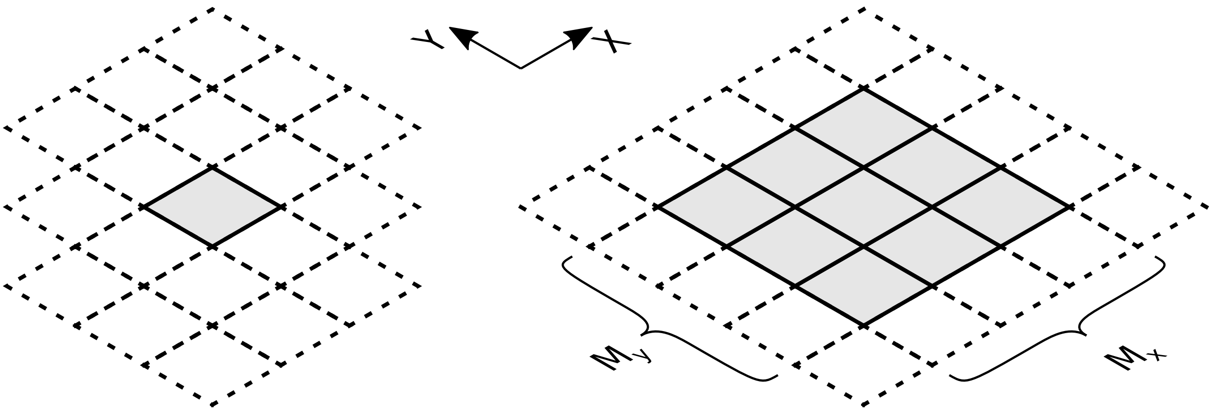
Prihodnje stanje neke celice $c_{x,y}$ na s koordinatami (x, y) je odvisno od trenutnega stanja pripadajoÄŊe okolice $n_{x,y}$ (slika 2.1). Tudi pri obliki okolice se bom omejil na pravokotnik velikosti $M_x \times M_y$. Åătevilo celic v okolici je $m = M_x \cdot M_y$. Na voljo je $|S|^m$ moÅ¿nih okolic.

$$n \in S^m \quad \text{in} \quad S^m = \{0, 1, \ldots, |S|^m - 1\} \tag{2.2}$$

Prostorski odnos med okolico in celico v prihodnjem stanju avtomata, ki jo ta okolica doloÄŊa, ni podrobno definiran. ObiÄŊajno se smatra, da je celica v sredini okolice, ampak za opisani algoritem to ni nujno pomembno. Moja implementacija algoritma celice, okolice in samo polje indeksira tako, da je zaÄŊetna ali referenÄŊna celica v spodnjem levem kotu.

Preslikava sedanje okolice $n^t_{x,y}$ v prihodnjo istoleÅ¿no celico $c^{t+1}_{x,y}$ je definirana s tranzicijsko funkcijo f, ki vsaki vrednosti okolice pripiÅąe vrednost celice:

$$c^{t+1}_{x,y} = f(n^t_{x,y}) \tag{2.3}$$



Slika 2.1: Celica $c_{x,y}$ in pripadajoÄŊa okolica $n_{x,y}$ z dimenzijama $M_x = M_y = 3$.

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

$$f^{-1}(c^t) = \{n^{t-1} \in S^m \mid f(n^{t-1}) = c^t\} \tag{2.4}$$

Dodaten pogoj za predslike polja celic je, da se morajo okolice sosednjih celic ujemati povsod, kjer se prekrivajo. Opazovanje prekrivanja okolic je tudi glavni element algoritmov za iskanje predslik.

Tranzicijsko funkcijo je moÅ¿no definirati s pravilom. Pravilo r je celo Åątevilo v S-iÅąkem Åątevilskem sestavu, kjer so cifre zaporedje vrednosti celic za vsako od $|S|^m$ moÅ¿nih okolic:

$$r = \sum_{n=0}^{n=|S|^m-1} |S|^n \cdot f(n) \tag{2.5}$$

$$r \in \{0, 1, \ldots |S|^{|S|^m} - 1\} \tag{2.6}$$

Vseh pravil je na voljo $|S|^{|S|^m}$. Pri 1D elementarnih CA ($m = 3$) je vseh pravil le 256, tako da so bila Å¿e vsa opazovana in opisana. Za 2D CA z okolico quad ($m = 4$) je pravil 65536, kar je nekje na meji, kar lahko raziskovalci podrobno pregledajo. Za 2D CA z Moorovo okolico ($m = 9$) je vseh pravil dosti preveÄŊ, da bi lahko pregledali posamiÄŊno.

Podana konstrukcija mreÅ¿e predslik in algoritem za izraÄŊun predslik omogoÄŊata uporabo bolj sploÅąne definicije. Za vsako celico je lahko definiran lasten nabor predslik $n^{t-1} \in S^m$, ki je na sploÅąno neodvisen od stanja celice. Ta posploÅąitev je uporabljena za konstrukcijo umetnih mreÅ¿ predslik, ki poudarjajo konkretne probleme, povezane s kompleksnostjo algoritma.

Za dano polje celic velikosti $N_x \times N_y$ in z odprtimi robovi, je moÅ¿no izraÄŊunati prihodnje stanje polja velikosti $(N_x - (M_x - 1)) \times (N_y - (M_y - 1))$. V primeru, da so robovi polja cikliÄŊno zaprti (polje v obliki torusa), pa je polje prihodnjega stanja enako veliko kakor polje sedanjega. Podobno velja za raÄŊunanje predslik. Za sedanje polje velikosti $N_x \times N_y$ je za odprte robove velikost polja predslik enaka $(N_x + (M_x - 1)) \times (N_y + (M_y - 1))$. Za cikliÄŊne robove pa sta velikosti enaki.

Poglavje 3

Konstrukcija mreÅ¿e predslik

MreÅ¿a predslik je grafiÄŊni konstrukt, ki omogoÄŊa upodobitev posameznih pojmov iz definicije CA, kakor loÄŊenih grafiÄŊnih elementov. Odnosi med temi elementi definirajo pravila, na katerih se gradijo algoritmi za iskanje predslik.

3.1 De Bruijnov diagram

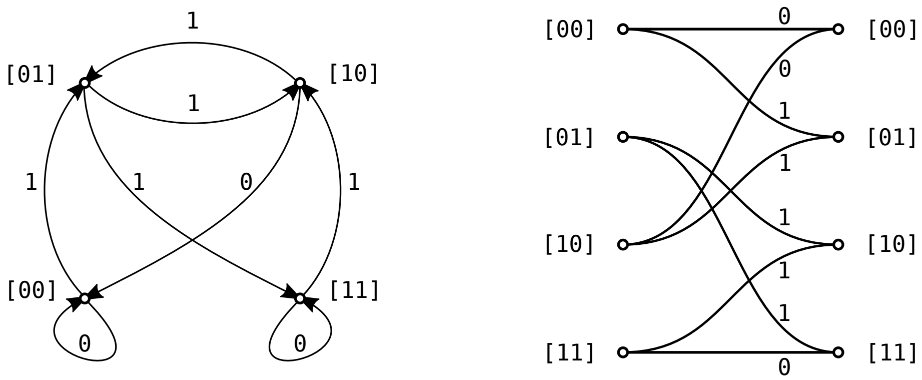
Osnovni element grafiÄŊne upodobitve je De Bruijnov diagram. V osnovi ta obravnava cikliÄŊne premike konÄŊnih zaporedij simbolov ter njihovo prekrivanje. VozliÅąÄŊa v diagramu so vsa moÅ¿na konÄŊna zaporedja, povezave med njimi pa definirajo, kako se ta zaporedja prekrivajo med seboj.

Pri 1D CA se problem neposredno preslika na De Bruijnov graf. McIntosh in njegova skupina uporabljajo za analizo De Bruijnove grafe neposredno. Sam pa sem razvil modificiran graf, kjer so vozliÅąÄŊa podvojena, in gredo poti vedno od originala proti dvojniku (slika 3.1). To omogoÄŊa veriÅ¿enje grafov in razÅąiritev osnovnega De Bruijnovega grafa. Medtem ko osnovni De Bruijnov graf opisuje okolico ene celice, veriÅ¿en graf opisuje verigo celic.

Za potrebe opisa 2D celiÄŊnih avtomatov je bila elementom De Bruijnovega grafa dodana nova dimenzija. Povezave med vozliÅąÄŊi se spremenijo v ploskve, vozliÅąÄŊa pa se spremenijo v robove ploskev. Elementi celiÄŊnega avtomata, ki se preslikajo v graf, so:

- nabor vseh moÅ¿nih **okolic** celice postane nabor vseh **ploskev** (slika 3.9)
- nabor **prekrivanj okolic v smeri 2D dimenzij** (slika 3.5) postane nabor **povezav**
- nabor **prekrivanj okolic v diagonalni smeri** (slika 3.7) postane nabor **vozliÅąÄŊ**

Elemente celiÄŊnega avtomata, kot so okolice in prekrivanja okolic, je potrebno indeksirati, tako da se lahko vsakemu elementu pripiÅąe unikatna Åątevna vrednost. Vsaki



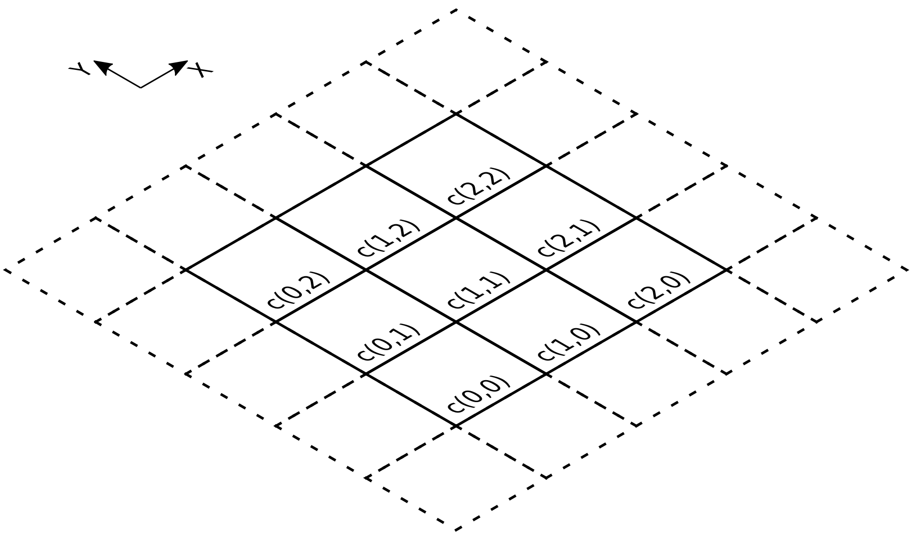
Slika 3.1: De Bruijnov graf za elementarni 1D CA, pravilo 110. Na levi je McIntosheva upodobitev, na desni pa moja, ki omogoÄŊa veriÅ¿enje.

okolici je pripisana zaporedna vrednost, ki je konstruirana kakor m-mestno Åątevilo v S-iÅąkem Åątevilskem sistemu (za podane primere dvojiÅąkem):

$$n = \sum_{\substack{x=0 \\ y=0}}^{\substack{x=M_x-1 \\ y=M_y-1}} |S|^{M_x y + x} \cdot c_{x,y} \tag{3.1}$$

$$n \in \{0, 1, \ldots, |S|^{M_x M_y} - 1\} \tag{3.2}$$

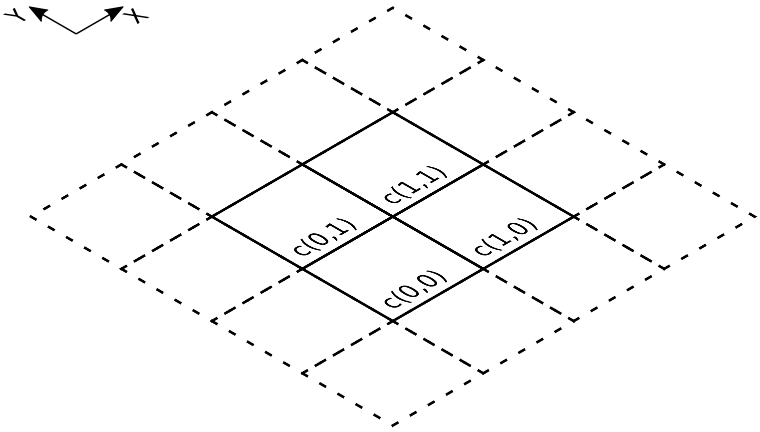
Cifre si sledijo od spodaj levo do zgoraj desno znotraj okolice (sliki 3.2 in 3.3).



Slika 3.2: Indeksiranje okolice celice z dimenzijama $M_x = M_y = 3$.

Celice se v smeri dimenzije X prekrivajo za ploskev velikosti $(M_x - 1) \times M_y$ (sliki 3.4 in 3.5). To ob indeksiranju da nabor:

$$o_{x^-} = \sum_{\substack{x=0 \\ y=0}}^{\substack{x=M_x-2 \\ y=M_y-1}} |S|^{(M_x-1)y + x} \cdot c_{x,y} \tag{3.3}$$



Slika 3.3: Indeksiranje okolice celice z dimenzijama $M_x = M_y = 2$.

$$o_{x^+} = \sum_{\substack{x=1\\y=0}}^{\substack{x=M_x-1\\y=M_y-1}} |S|^{(M_x-1)y+x} \cdot c_{x,y} \tag{3.4}$$

$$o_x \in \{0, 1, \ldots, |S|^{(M_x-1)M_y} - 1\} \tag{3.5}$$

Celice se v smeri dimenzije Y prekrivajo za ploskev velikosti $M_x \times (M_y - 1)$ (sliki 3.4 in 3.5). To ob indeksiranju da nabor:

$$o_{y^-} = \sum_{\substack{x=0\\y=0}}^{\substack{x=M_x-1\\y=M_y-2}} |S|^{M_x y+x} \cdot c_{x,y} \tag{3.6}$$

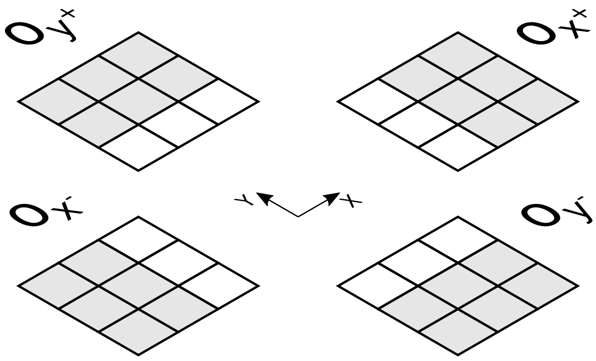
$$o_{y^+} = \sum_{\substack{x=0\\y=1}}^{\substack{x=M_x-1\\y=M_y-1}} |S|^{M_x y+x} \cdot c_{x,y} \tag{3.7}$$

$$o_y \in \{0, 1, \ldots, |S|^{M_x(M_y-1)} - 1\} \tag{3.8}$$

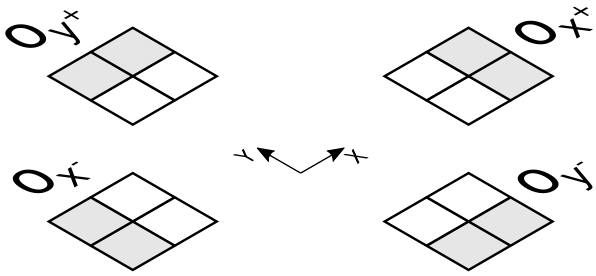
Celice se v diagonalni smeri prekrivajo za ploskev velikosti $(M_x - 1) \times (M_y - 1)$ (sliki 3.6 in 3.7). To ob indeksiranju da nabor:

$$o_{x^-y^-} = \sum_{\substack{x=0\\y=0}}^{\substack{x=M_x-2\\y=M_y-2}} |S|^{(M_x-1)y+x} \cdot c_{x,y} \tag{3.9}$$

$$o_{x^+y^-} = \sum_{\substack{x=1\\y=0}}^{\substack{x=M_x-1\\y=M_y-2}} |S|^{(M_x-1)y+x} \cdot c_{x,y} \tag{3.10}$$



Slika 3.4: Prekrivanje okolic sosednjih celic v smeri dimenzij X in Y, za velikost okolice $M_x = M_y = 3$. Okolice se prekrivajo v 6 celicah od 9.



Slika 3.5: Prekrivanje okolic sosednjih celic v smeri dimenzij X in Y, za velikost okolice $M_x = M_y = 2$. Okolice se prekrivajo v 2 celicah od 4.

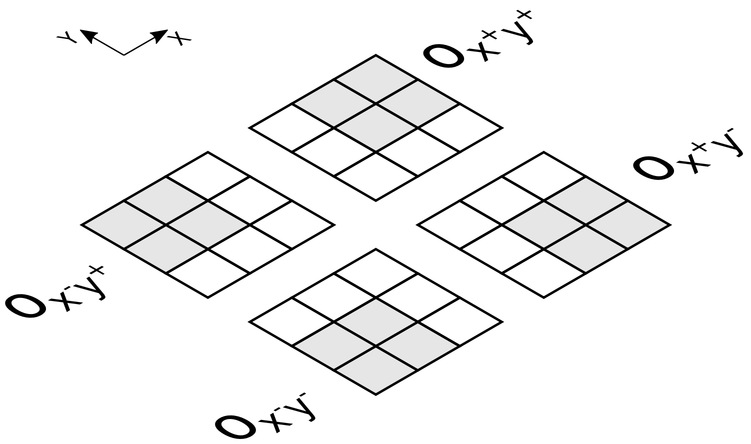
$$o_{x^-y^+} = \sum_{\substack{x=0\\y=1}}^{\substack{x=M_x-2\\y=M_y-1}} |S|^{(M_x-1)y+x} \cdot c_{x,y} \tag{3.11}$$

$$o_{x^+y^+} = \sum_{\substack{x=1\\y=1}}^{\substack{x=M_x-1\\y=M_y-1}} |S|^{(M_x-1)y+x} \cdot c_{x,y} \tag{3.12}$$

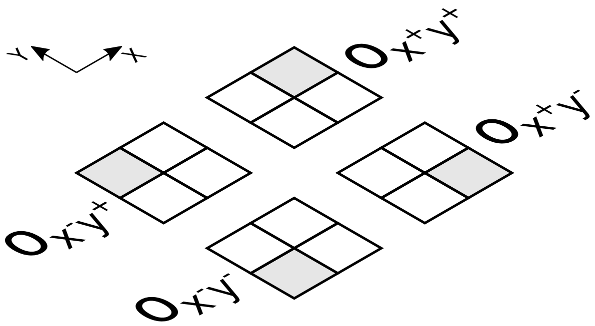
$$o_{xy} \in \{0, 1, \ldots, |S|^{(M_x-1)(M_y-1)} - 1\} \tag{3.13}$$

Graf predslik je postavljen na kvadrat, ki predstavlja okolico n. Nad vsakim vogalom je nabor vozliÅąÄŊ, ki predstavljajo nabor vogalnih prekrivanj $o_{x^-y^-}$, $o_{x^+y^-}$, $o_{x^-y^+}$ in $o_{x^+y^+}$. VozliÅąÄŊi sosednjih vogalov sta povezani, ÄŊe sta del iste konfiguracije celic, ki predstavlja povezavo. Na primer, pri okolici quad povezava za prekrivanje okolic `[01]` povezuje vozliÅąÄŊi `[0]` in `[1]` (slika 3.8 levo).

Nastali graf (slika 3.8 desno) ima poleg vozliÅąÄŊ in povezav med njimi tudi ploskve. Ploskve bi v teoriji grafov opisali kakor zanke v grafu, z dodano omejitvijo, da mora vsako vozliÅąÄŊe ali povezava v zanki pripadati drugemu prekrivanju okolic.



Slika 3.6: Prekrivanje okolic sosednjih celic v diagonalni smeri, za velikost okolice $M_x = M_y = 3$. Okolice se prekrivajo v 4 celicah od 9.



Slika 3.7: Prekrivanje okolic sosednjih celic v diagonalni smeri, za velikost okolice $M_x = M_y = 2$. Okolice se prekrivajo v eni celici od 4.

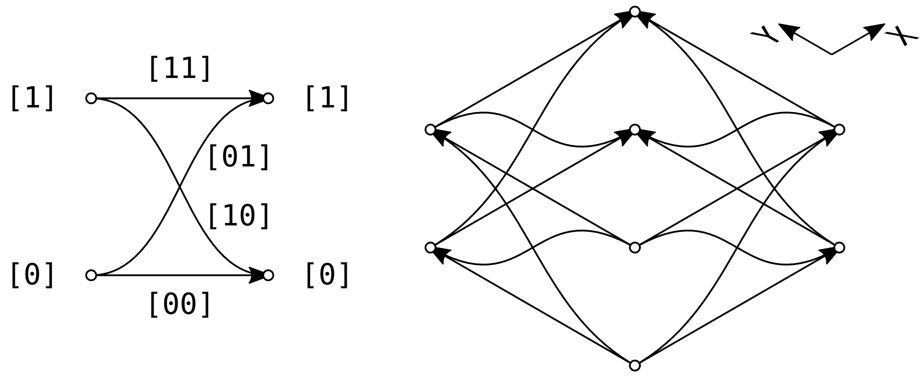
3.2 MreÅ¿a

Diagrame, ki opisujejo preteklost posamezne celice, je moÅ¿no sestaviti v mreÅ¿o, ki opisuje polje veÄŊ celic. Nabor predslik celotnega polja je ekvivalenten naboru vseh *zveznih* ploskev, ki prekrivajo celotno polje (zahteva po prekrivanju okolic) in ki jih je moÅ¿no sestaviti iz naborov ploskev diagramov posameznih celic (zahteva po ujemanju s tranzicijsko funkcijo).

Preslikava iz zvezne ploskve v mreÅ¿i predslik v konfiguracijo polja celic predslike je enoliÄŊna. NajlaÅ¿je je razumeti preslikavo za okolico velikosti 3×3. Od vsakega odseka ploskve za posamezno celico vzamemo centralno celico, na koncu pa dodamo Åąe za eno celico Åąirok rob okoli celotne ploskve.

Podani primeri uporabljajo binarni CA z okolico quad. Za ta CA je velikost diagonalnega prekrivanja okolic ena sama celica (slika 3.7); poslediÄŊno ima nabor vozliÅąÄŊ le dve vrednosti, ki neposredno predstavljajo vrednosti celic v predsliki (slika 3.10). Nabor okolic/ploskev pa obsega 16 kombinacij (slika 3.9).

Pravilnost razÅąiritve diagrama ene celice v mreÅ¿o lahko dokaÅ¿emo z indukcijo. Dokazati Å¿elimo, da je neka konfiguracija celic predslika dane sedanjosti, *ÄŊe in samo ÄŊe* je ta ekvivalentna zvezni ploskvi v mreÅ¿i predslik.



Slika 3.8: MreÅ¿a ene celice za binarni CA z okolico quad $M_x = M_y = 2$.

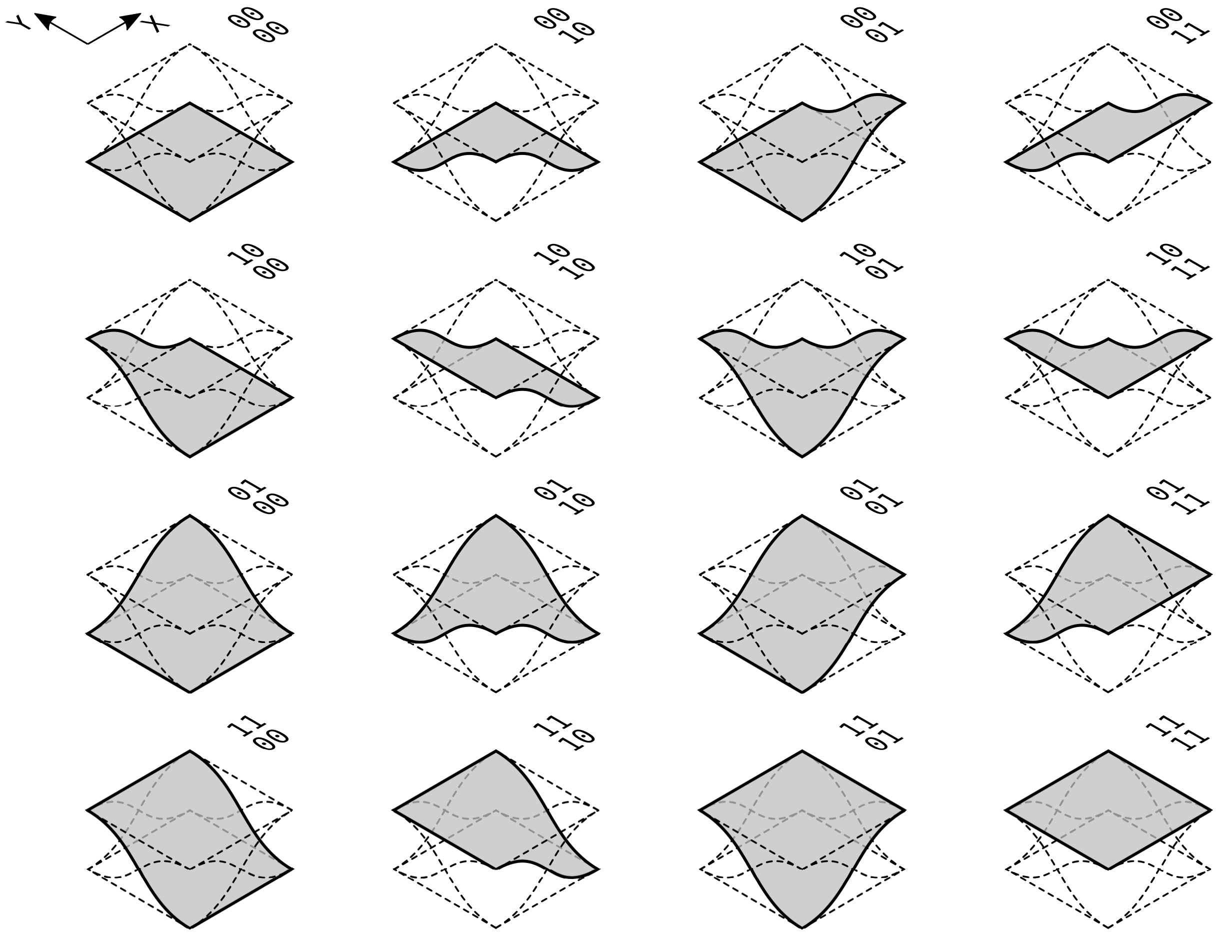
Prvi element: Iz definicije velja, da je za eno samo celico v mreÅ¿i nabor ploskev enak naboru vseh predslik. **Naslednji element:** ObstojeÄŊi mreÅ¿i predslik dodamo novo celico. Ploskev iz nabora dodane celice se zvezno veÅ¿e s ploskvijo iz obstojeÄŊega nabora zveznih ploskev, ko se z njo ujema v robu (vozliÅąÄŊema in povezavi med njima). Robovi ploskev se ujemajo natanko v primeru, ko se ujemajo prekrivanja okolic. Temu je tako, ker so indeksi vozliÅąÄŊ in povezav ekvivalentni vrednosti prekrivanj okolic.

3.3 Robni pogoji

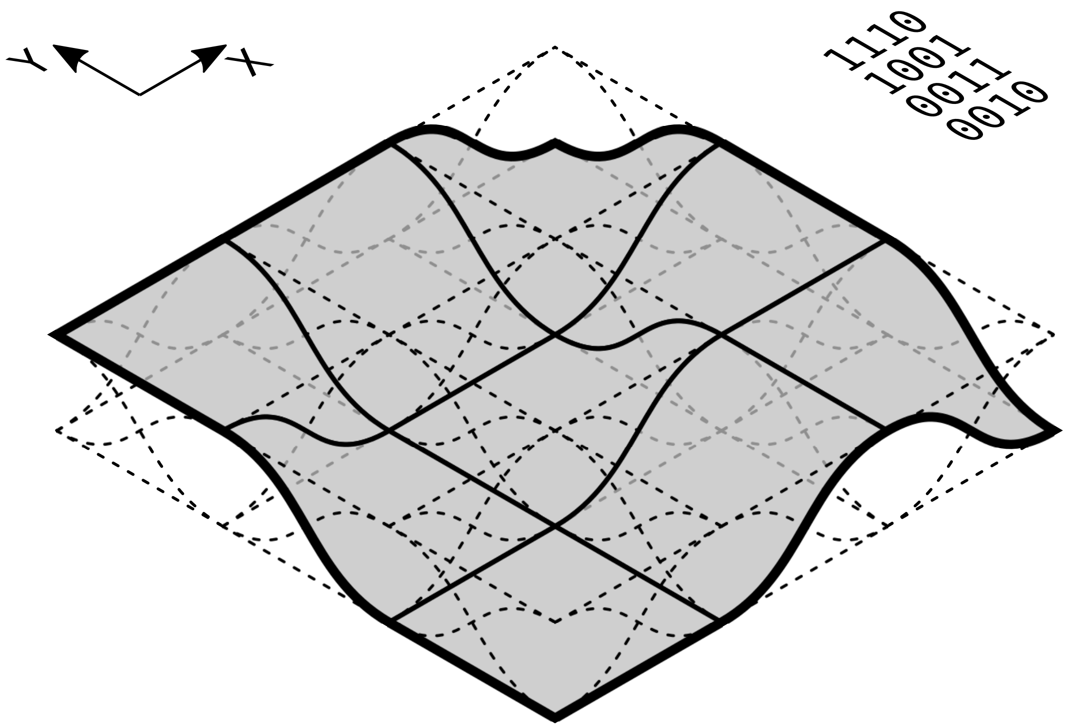
Pri 1D CA so robni pogoji definirani na dveh koncih, ki omejujejo konÄŊno Åątevilo celic na neskonÄŊni premici. ÄŇe je 1D CA definiran kakor poltrak, je robni pogoj samo eden. Robni pogoj definira, katere okolice (vozliÅąÄŊa pri mreÅ¿i predslik za 1D CA) in s kakÅąnimi uteÅ¿mi so na voljo ob robu. Lahko si jih predstavljamo tudi kakor vpliv neskonÄŊnega poltraka celic, ki sega izven roba opazovane konfiguracije.

Pri 2D CA je robni pogoj definiran kakor uteÅ¿ sklenjene poti okoli ploskve opazovane konfiguracije celic. V mreÅ¿i predslik za 2D CA povezave med vozliÅąÄŊi definirajo rob ploskve (slika 3.10). Vsaka zvezna ploskev v mreÅ¿i ima zvezen rob. Robni pogoj doloÄŊa, kako je ta ploskev obravnavana, oziroma, ali je sploh obravnavana. Ker uporabna vrednost sploÅąnega robnega pogoja Åąe ni znana, in bi sploÅąnost izrazito poveÄŊala zahtevnost algoritma za iskanje predslik, so tukaj vsi robovi obravnavani enako, z uteÅ¿jo ena. Temu bomo rekli odprti rob, ker ta ne definira nobenih omejitev, katere zvezne ploskve v mreÅ¿i predslik so dovoljene in katere ne.

Obstaja Åąe en enostaven robni pogoj, ki je definiran za cikliÄŊno sklenjena konÄŊna polja CA. Ta tip robnega pogoja tukaj ne bo obravnavan, ker Åąe dodatno poveÄŊa kompleksnost algoritmov. Potrebno bi bilo namreÄŊ opraviti celoten postopek iskanja predslik za vsak zaÄŊetni rob posebej. Zvezna ploskev je predslika v cikliÄŊnem polju le, ÄŊe se njen zaÄŊetni in konÄŊni rob ujemata. Pojma zaÄŊetnega in konÄŊnega roba sta opisana v poglavju o algoritmu.



Slika 3.9: Nabor ploskev za vseh 16 moÅ¿nih vrednosti okolic za binarni CA z okolico quad $M_x = M_y = 2$.



Slika 3.10: MreÅ¿a velikosti $N_x = 3$ in $N_y = 3$ za binarni CA z okolico quad. Poudarjena je ena zvezna ploskev in njen zvezen rob. Konfiguracija pripadajoÄŊe predslike je izpisana.

Poglavje 4

Algoritem za Åątetje in izpis predslik

Pri 1D CA ima algoritem za Åątetje predslik linearno $O(N)$ procesno in pomnilniÅąko zahtevnost. PosploÅąeno to pomeni, da se vsaka celica pojavi v izraÄŊunu samo enkrat. Dejansko ima vsak algoritem tudi logaritmiÄŊno komponento, saj Åątevilo bitov, potrebnih za zapis Åątevcev, raste logaritmiÄŊno s Åątevilom celic. Algoritem za izpis predslik ima neizogibno eksponentno kompleksnost $O(C^N)$, saj maksimalno in povpreÄŊno Åątevilo predslik raste eksponentno v odvisnosti od Åątevila celic.

Za 2D CA se je izkazalo, da obstajajo problemi, ki niso reÅąljivi z linearno kompleksnostjo. Prikazani algoritem ima maksimalno eksponentno kompleksnost v odvisnosti od posamezne dimenzije $O(S^{N_x}S^{N_y})$. Za sedaj je Åąe odprta moÅ¿nost za obstoj algoritma s polinomsko kompleksnostjo, ki je nekje med linearno in eksponentno.

Opisani algoritem za Åątetje predslik razdeli polje celic na vrstice v dimenziji X. Delitev po stolpcih v dimenziji Y bi bila ekvivalentna, tako da je to arbitrarna odloÄŊitev. S staliÅąÄŊa procesne zahtevnosti je najbolje izbrati krajÅąo dimenzijo. Na podlagi zunanjega robnega pogoja algoritem najprej poiÅąÄŊe nabor predslik za prvo vrstico. Nabor predslik izraÄŊuna kakor uteÅ¿i za robove na nasprotni strani od zaÄŊetne. Seznam robov in njihove uteÅ¿i uporabi kakor vhodni robni pogoji za naslednjo vrstico. Robne uteÅ¿i, izraÄŊunane za zadnjo vrstico, predstavljajo Åątevilo vseh predslik.

Algoritem za izpis predslik je nadaljevaje algoritma za Åątetje. ZaÄŊne z znanim Åątevilom predslik, ki ga je dalo Åątetje, in izpisuje predslike po vrsticah v obratni smeri, kakor je potekalo Åątetje (od zadnje do prve vrstice). Pridobljen seznam predslik je sortiran. Smer sortiranja je odvisna od smeri procesiranja. Smer procesiranja je moÅ¿no po potrebi spremeniti.

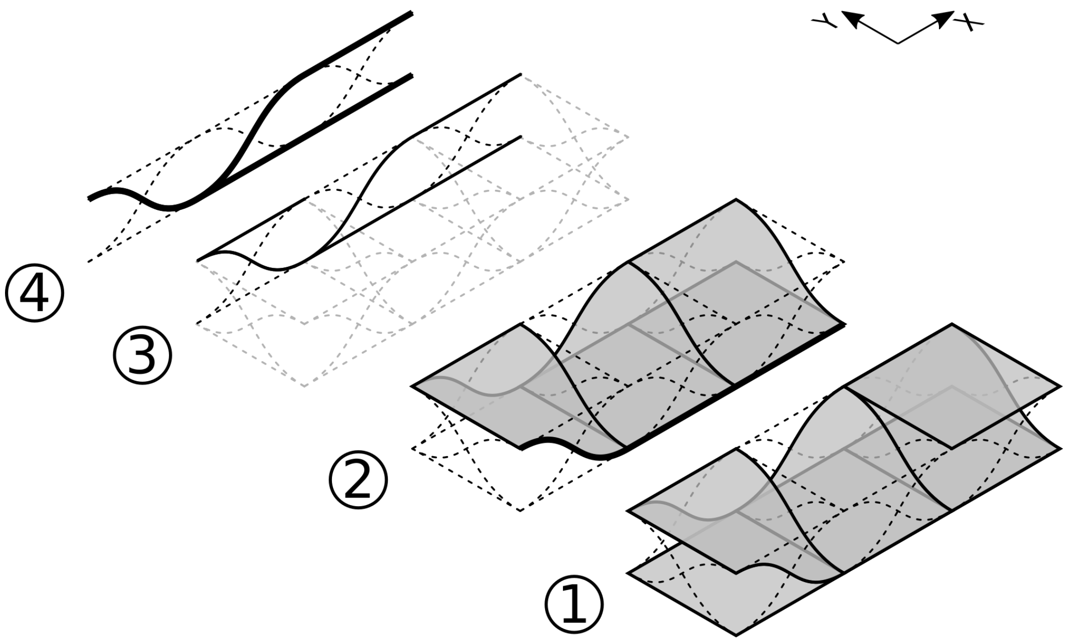
4.1 Procesiranje v eni dimenziji

Procesiranje se zaÄŊne z 1D nizom celic (vrstico). Vsaki celici v nizu pripada lastna mreÅ¿a predslik, eno dimenzionalnemu nizu celic slediÄŊno pripada povezan niz mreÅ¿ predslik (slika 4.1 mreÅ¿a **1**). Vsak segment mreÅ¿e v nizu ima svoj nabor veljavnih okolic, ki je definiran s tranzicijsko funkcijo (ali pa je posploÅąeno poljuben nabor). Na zaÄŊetku se ploskve okolic sosednjih celic Åąe ne povezujejo v zvezno ploskev ÄŊez celoten niz. IzloÄŊiti je potrebno vse ploskve okolic, ki se ne povezujejo z okolicami svojih sosednjih celic.

Ker se vsaka vrstica povezuje s predhodno in naslednjo vrstico, je potrebno upoÅątevati tudi zveznost ploskev na prehodu med vrsticami. Ta povezava med vrsticami je prerez opazovane ploskve na meji med vrsticama. Vsako ploskev se obravnavana posebej in prerez se izrazi kakor robni pogoj. ZaÄŊetni robni pogoj za vsako vrstico je nabor poti med vozliÅąÄŊi na zaÄŊetku vrstice. Vsaka pot je obravnavana posebej (slika 4.1, mreÅ¿a **2**, odebeljena pot). ZaÄŊetni robni pogoj je ekvivalenten nizu prekrivanja okolic o_{y^-}.

ZaÄŊetni robni pogoj se aplicira tako, da se izloÄŊi iz obravnave vse ploskve, ki nimajo skupnega roba z robno potjo (slika 4.1, prehod iz mreÅ¿e **1** v mreÅ¿o **2**). Po tem koraku, so vse ploskve definirane v dveh od Åątirih vozliÅąÄŊ. Do nezveznosti lahko prihaja le Åąe na robu, nasprotnem zaÄŊetnemu robu.

Problem se reducira iz 3D mreÅ¿e in procesiranja ploskev v 2D mreÅ¿o, kjer se procesirajo poti med vozliÅąÄŊi (slika 4.1, prehod iz mreÅ¿e **2** v mreÅ¿o **3**). To je problem ekvivalenten iskanju predslik (zvenih poti v grafu) za 1D CA, za kar se uporabi algoritem, opisan v [16]. Rezultat so poti na prerezu med ploskvami (slika 4.1, mreÅ¿a **4**, konÄŊni robni pogoj je odebeljen). KonÄŊni robni pogoj je ekvivalenten nizu prekrivanja okolic o_{y^+}.



Slika 4.1: MreÅ¿a vrstice velikosti $N_x = 3$ za binarni CA z okolico quad. Nabor okolic/ploskev za posamezno celico je arbitraren. Izbran je tako, da poudari korake algoritma.

Po procesiranju nabora vseh zaÄŊetnih robnih pogojev nastane nabor vseh konÄŊnih robnih pogojev. Ta nabor se v naslednjem koraku uporabi kakor zaÄŊetni robni pogoj za naslednjo vrstico.

Nabor vseh moÅ¿nih poti na meji vrstice raste eksponentno z dolÅ¿ino vrstice. Åătevilo vseh poti se izraÄŊuna iz Åątevila celic, ki se prekrivajo med okolicama dveh vrstic. Za deljenje po vrsticah v dimenziji X je ta nabor velikosti $|S|^{(M_y-1)(M_x-1+N_x)}$. PoslediÄŊno kompleksnost algoritma raste eksponentno z dolÅ¿ino vrstice $O(C^{N_x})$.

4.2 Procesiranje v drugi dimenziji

V drugi dimenziji se procesira zaporedje vrstic od prve do zadnje (slika 4.2). Vsaka vrstica se zaÄŊne in konÄŊa z robnim pogojem. Ti robni pogoji so prerezi ploskev v celotni mreÅ¿i predslik. Vsako vrstico je potrebno v drugi dimenziji procesirati le enkrat, kar pomeni, da je raÄŊunska zahtevnost linearno odvisna od Åątevila vrstic $O(N_y)$.

Vsakemu robnemu pogoju je pripisana uteÅ¿. ZaÄŊetni robovi za prvo vrstico imajo pripisano uteÅ¿ $w = 1$. Na sploÅąno se lahko veÄŊ zaÄŊetnih robov preslika v isti konÄŊni rob ene vrstice. UteÅ¿, pripisana konÄŊnemu robu dane vrstice, je vsota uteÅ¿i vseh zaÄŊetnih robnih pogojev, ki se preslikajo vanj. UteÅ¿ tako predstavlja Åątevilo vseh predslik za dano in prejÅąnje vrstice, ki se konÄŊajo z opazovanim robom. Robovi, s katerimi se ne konÄŊa nobena od predslik, dobijo uteÅ¿ $w = 0$.

Za tem, ko so procesirane vse vrstice, je znan konÄŊni rob nabora vseh zveznih ploskev na celotni mreÅ¿i predslik. Vsota uteÅ¿i vseh konÄŊnih robov daje Åątevilo predslik celotne mreÅ¿e.

Z uporabo Boolove algebre namesto operacij mnoÅ¿enja in seÅątevanja je moÅ¿no poenostaviti iskanje predslik samo na njihov obstoj za opazovani robni pogoj. UteÅ¿i postanejo binarne vrednosti $w \in \{0, 1\}$.

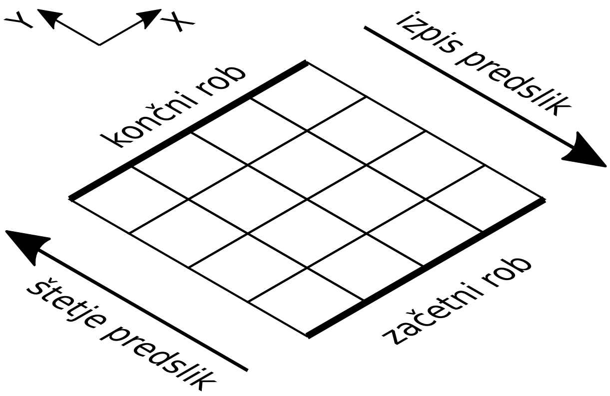
4.3 Izpis predslik

Ni nujno, da se vsak vrstiÄŊni zaÄŊetni pogoj preslika v nabor konÄŊnih pogojev. MoÅ¿no je, da nobena pot na konÄŊnem robu ne zadoÅąÄŊa zaÄŊetnemu pogoju in mreÅ¿i predslik. Torej po prejÅąnjih korakih Åąe ni toÄŊno doloÄŊeno, katere ploskve se zdruÅ¿ujejo v zvezno celoto in katere ne. MoÅ¿ne so ploskve, ki prekrivajo polje samo do neke vrstice in ne naprej.

Procesiranje po drugi dimenziji poteka v obratni smeri kakor pri Åątetj; zaÄŊne pri zadnji vrstici in konÄŊa pri prvi vrstici (slika 4.2). Z vsakim korakom se izloÄŊa Åąe preostale slepe poti iz mreÅ¿e predslik. Za vsako vrstico je potrebno ponovno reÅąiti 1D problem, le da sta zaÄŊetni in konÄŊni rob zamenjana. ZaÄŊetni rob je zaporedje prekrivanja okolic o_{y^+} in konÄŊni rob zaporedje prekrivanja okolic o_{y^-}.

Hkrati je moÅ¿no Åąe izpisovati predslike. Å¡e na zaÄŊetku procesiranja v obratni smeri je znano Åątevilo vseh predslik, kar omogoÄŊa rezervacijo pomnilnika in inicializacijo seznama predslik. KonÄŊne robne poti in njihove uteÅ¿i se uporabijo za popis tekoÄŊe vrstice celic v predslikah. Indeks robne poti (doloÄŊeno zaporedje prekrivajoÄŊih okolic) da vrednost celic v vrstici predslike. UteÅ¿i, izraÄŊunane v smeri Åątetja, pa dajo Åątevilo predslik, ki jih je potrebno popisati z dano vrednostjo za vrstico celic.

Z vsakim korakom v obratni smeri se popiÅąe nova vrstica za vsako od preÅątetih predslik. Ko pride algoritem spet do zaÄŊetne vrstice, so vse predslike popisane. Hkrati so iz mreÅ¿e predslik izloÄŊene vse slepe poti; ostanejo le Åąe ploskve, ki tvorijo zvezne ploskve na celotnem polju celic.



Slika 4.2: Potek smeri procesiranja pri algoritmu za Åątetje in izpis predslik.

Algoritem tukaj ni podrobno opisan, je pa preprosta razÅąiritev algoritma uporabljenega za 1D CA, opisanega v [16]. Za podrobnosti je na voljo izvorna koda [12].

4.4 NezmoÅ¿nost procesiranja z linearno zahtevnostjo

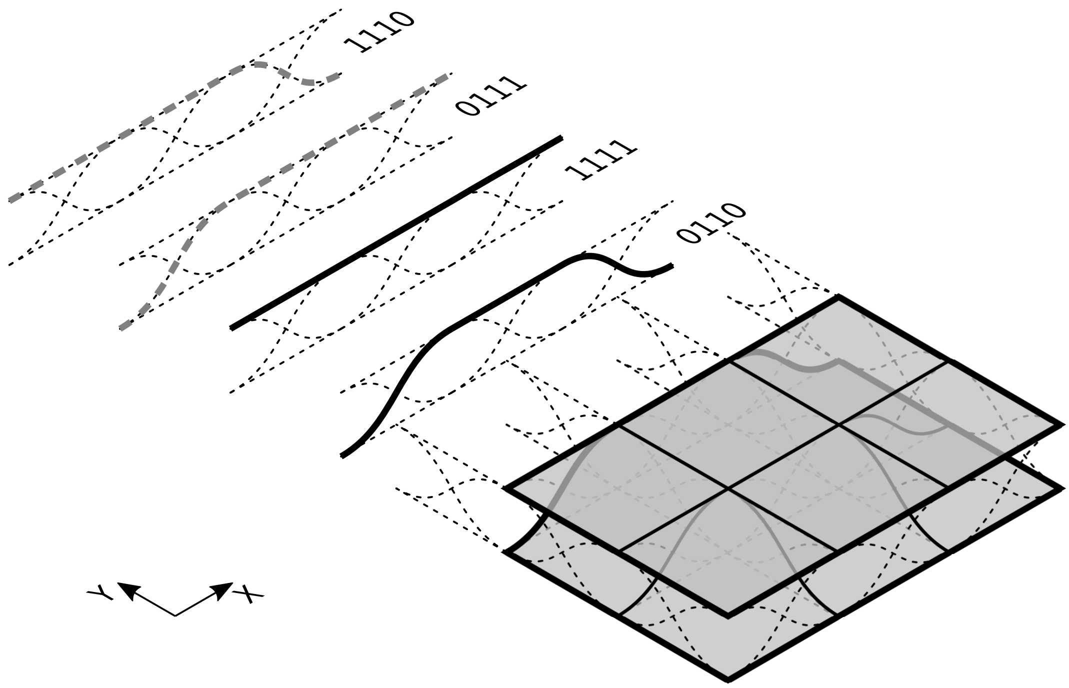
Podan je primer, ki kaÅ¿e, zakaj procesiranje z linearno zahtevnostjo ni mogoÄŊe.

Cilj raziskovalnega dela za to raziskovalno nalogo je bil poiskati uÄŊinkovit algoritem za iskanje predslik 2D CA. Na podlagi izkuÅąenj z 1D CA sem optimistiÄŊno priÄŊakoval, da bo moÅ¿no problem reÅąiti v linearnem ÄŊasu.

Algoritem v linearnem ÄŊasu bi vsako celico obravnaval le enkrat; na sploÅąno bi bilo Åątevilo obravnav majhna konstanta (4-krat, ÄŊe se izvaja procesiranje skozi mreÅ¿o predslik v 4 smereh/prehodih). Tak algoritem predpostavlja, da je moÅ¿no celoten nabor zaÄŊetnih robnih pogojev upoÅątevati hkrati v enem samem prehodu. V koraku **2** pri procesiranju vrstice (slika 4.1) bi se izloÄŊilo le ploskve, ki ne ustrezajo nobenemu od zaÄŊetnih robnih pogojev. Po nekaj poizkusih sem ugotovil, da tak algoritem ne deluje pravilno. Vsaj nekatere poti iz nabora je potrebno upoÅątevati loÄŊeno.

V podanem primeru (slika 4.3) se prvi dve vrstici zakljuÄŊita z naborom dveh konÄŊnih poti **0110** in **1111**. ÄŇe bi ti dve poti uporabili hkrati, kakor zaÄŊetni robni pogoj za naslednjo vrstico, bi dobili enak rezultat, kakor ÄŊe bi bili del robnega pogoja tudi poti **0111** in **1110**. To pa zato, ker algoritem, ki poti ne obravnava loÄŊeno, ne more vedeti, kje se je pot zaÄŊela, da bi jo lahko tudi pravilno zakljuÄŊil. PoslediÄŊno linearen algoritem lahko vzame zaÄŊetek ene od poti in ga na vozliÅąÄŊu, skupnem obema potema, nadaljuje po drugi poti. Iz mreÅ¿e je razvidno, da ti dve poti nista preseka neke zvezne ploskve. Skratka, hkratno obravnavanje celotnega nabora robnih pogojev ni moÅ¿no, kar pomeni, da je Åątevilo obravnav neke celice odvisno od Åątevila poti in poslediÄŊno od eksponenta Åątevila celic v vrstici.

MoÅ¿no je, da obstajajo drugaÄŊne optimizacije algoritma, ki lahko zmanjÅąajo procesno kompleksnost.



Slika 4.3: Ovira za procesiranje z linearno zahtevnostjo. Veljavni robni pogoj je odebeljen, neveljavni pa ÄŊrtkan.

Poglavje 5

Primerjava z znanimi algoritmi

ObstojeÄŊi algoritmi za predslike 2D CA se osredotoÄŊajo izkljuÄŊno na GoL. NajveÄŊ algoritmov je namenjenih iskanju stanj GoE, skratka, preverjajo le obstoj predslik in jih ne Åątejejo ali izpisujejo. Tukaj opisani algoritem je zamiÅąljen bolj univerzalno za poljuben 2D CA. Vendar univerzalnost v praksi ni ravno pomembna, saj izven raziskovalcev GoL ni dosti zanimanja za teoretiÄŊne probleme, kakor je iskanje predslik.

5.1 Iskanje stanj GoE za GoL

Conway je leta 1970 predstavil CA po imenu GoL. Å¡e leta 1971 sta Roger Banks in Steve Ward predstavila prvo stanje GoE velikosti 9×33 celic. Odkritje je bilo objavljeno v [27]. Kasneje je Don Woods [27, 28] s pomoÄŊjo raÄŊunalnika dokazal, da je stanje res GoE.

Woodsov algoritem, tako kakor moj, deluje na vrsticah. ZaÄŊne z eno vogalno celico, katere nabor predslik je kar nabor okolic, ki se preslikajo v stanje celice. V naslednjem koraku opazuje sosednjo celico v vrstici. Za vse kombinacije okolic dodane celice in obstojeÄŊi nabor predslik preveri, ÄŊe se ujemajo, kjer se okolice prekrivajo. ÄŇe se ujemajo, doda sestavljeno predsliko v nabor, sicer predslika izpade iz nabora. Ko pride algoritem do konca vrstice, nadaljuje v naslednji vrstici, le da je sedaj prekrivanje med naborom predslik in okolico dodane celice drugaÄŊno (veÄŊ sosednjih celic namesto ene). Tako algoritem nadaljuje do konca celotne konfiguracije. ÄŇe je konÄŊni nabor predslik prazen, je opazovana konfiguracija GoE. Nabor predslik in s tem poraba pomnilnika in procesnega ÄŊasa ostajajo relativno majhni v primerjavi z naborom vseh stanj. Temu je tako, ker imajo v praksi tudi deli konfiguracije GoE malo predslik. Algoritem bi bil za konfiguracije z dosti predslikami veliko bolj pomnilniÅąko in ÄŊasovno potraten.

Nicolay Beluchenko je poiskal veÄŊje Åątevilo stanj GoE velikosti 11×11. Sicer ne poznam vseh njegovih pristopov, ampak vsaj v enem primeru je uporabljal podoben algoritem kakor Woods. ZaÄŊel je z osrednjo celico in dodajal nove celice v spirali okoli osrednje. Za vrednost dodane celice je uporabil tisto, ki je dala manjÅąi nabor predslik. Algoritem zakljuÄŊi, ko z novo dodano celico ni veÄŊ moÅ¿no tvoriti predslik.

Leta 1974 je Duparc [7, 8] razvil drugaÄŊen algoritem, ter z njim poiskal stanja GoE velikosti 6×122 in 6×117. Algoritem uporablja teorijo konÄŊnih avtomatov in regularnih jezikov, ki je v osnovi namenjena 1D sistemom. Duparc je celice iz vrstice 2D polja zdruÅ¿il v simbole regularnega jezika, zaporedje veÄŊ vrstic pa predstavlja besedo. Duparcov algoritem ne obravnava vseh stanj vrstice enakovredno, temveÄŊ se osredotoÄŊa na vrstice, ki zmanjÅąujejo nabor predslik. Obravnavanje vseh stanj vrstic bi namreÄŊ preseglo pomnilniÅąke sposobnosti takratne in sedanje strojne opreme Å¿e za kratke vrstice. DokaÅ¿e le, da za vrstice dolÅ¿ine 1 ne obstaja stolpec, ki bi bil stanje GoE. Originalni ÄŊlanek je v francoÅąÄŊini, tako da lahko algoritem opiÅąem le na podlagi tega, kako ga opisujejo drugi.

V zadnjih letih podoben algoritem uporablja Steven Eker (CSL, SRI International, California, USA). Dokazal je, da za pravokotne konfiguracije viÅąine 2 in 3 ne obstajajo stanja GoE. Dokaz za konfiguracije viÅąine 4 z danim algoritmom Åąe vedno presega sposobnosti sodobne strojne opreme. Åătevilo stanj avtomata, ki opisuje jezik GoE, namreÄŊ raste eksponentno z viÅąino. Elker je poiskal stanja GoE velikosti 9×11, 8×12 in 5×83. Na Å¿alost raziskovalcu delodajalec prepoveduje objavo rezultatov in podrobnosti, dokler ni opravljen interni pregled. Tako imamo za sedaj na voljo le zgoraj navedene podatke.

DrugaÄŊen algoritem uporablja Marijn Heule [9]. S sodelavci je zapisal vsa prekrivanja za doloÄŊeno stanje v obliki binarnih enaÄŊb. Te je nato predal orodju za reÅąevanje SAT problemov. ÄŇe reÅąitev ne obstaja, pomeni, da je stanje GoE. Uporabljena je dodatna predpostavka, da bo reÅąitev zrcalno simetriÄŊna v obeh dimenzijah, kar obÄŊutno zmanjÅąa velikost problema. Algoritem je Åąe dodatno optimiziran za iskanje stanj GoE in omogoÄŊa pregled veÄŊjega Åątevila podobnih stanj. Ob spremembi vrednosti ene same celice omogoÄŊa uporabo delnih rezultatov prejÅąnjega izraÄŊuna. SAT metode ne poznam dobro, tako da ne morem primerjati procesne zahtevnosti.

5.2 Iskanje predslik GoL na sploÅąno

Erlan [5] je predstavil igro, pri kateri se za dano stanje GoL iÅąÄŊe predslike. To dejansko ni implementacija algoritma za iskanje predslik, je pa vzpodbudil druge k iskanju takega algoritma.

Spletna stran `kaggle.com` je pripravila tekmovanje [1], kjer so morali kandidati reÅąevati problem iskanja predslik. Podali so nabor stanj, za katere je treba poiskati predslike, in nabor uÄŊnih sekvenc. PriÄŊakovali so reÅąitev, ki temelji na strojnem uÄŊenju ali optimizaciji, torej fokus ni bil na eksaktni reÅąitvi.

Pri iskanju implementacij algoritma, sem naÅąel Atabot [3], CeliÄŊni kronometer [4] in program baziran na SAT metodi [21]. Cilj teh algoritmov sicer je iskanje predslik, ampak iskanje ni sistematiÄŊno in cilj ni nabor vseh moÅ¿nih predslik. Algoritmi se tudi posluÅ¿ujejo doloÄŊene hevristike pri iskanju.

5.3 Polni algoritmi za Åątetje in izpis predslik

Opisani algoritmi za iskanje stanj GoE, na primer Woods, sicer omogoÄŊajo iskanje vseh predslik, ampak to ni njihov glavni namen. Optimizirani so le za stanja GoE, kjer je predslik malo tudi za samo del konfiguracije GoE. Predvsem niso optimizirani za sploÅąen primer, kjer je lahko predslik veliko, poslediÄŊno so potratni s porabo pomnilnika. Morda bi bilo moÅ¿no optimizirati porabo pomnilnika, tako da bi algoritem hranil le tekoÄŊe razlike med predslikami, namesto da jih hrani v celoti. Nekaj podobnega poÄŊne moj algoritem, ko Åąteje le robove, in ne hrani celotnih predslik.

NaÅąel sem le eno aplikacijo, ki je namenjena Åątetju in izpisu vseh predslik. Bickford [2] prav tako kakor ostali procesira konfiguracijo po vrsticah, znotraj vrstice pa lahko uporablja razliÄŊne algoritme (Woods, Duparc). Bickford opiÅąe tudi algoritem, ki namesto procesiranja po vrsticah zdruÅ¿uje predslike kvadratnih podsklopov konfiguracij. Na Å¿alost poda primerjavo med razliÄŊnimi algoritmi le kakor procesni ÄŊas za posamezno implementacijo, ne pa tudi bolj sploÅąne ocene maksimalne in povpreÄŊne procesne/pomnilniÅąke zahtevnosti.

NaÅąel sem Åąe aplikacijo, napisano v jeziku APL [10]. Dokumentacija te aplikacije ni dovolj jasna, da bi bilo oÄŊitno, kaj toÄŊno poÄŊne. Iz primera pa sklepam, da je sposobna poiskati vse predslike.

Poglavje 6

Sklepne ugotovitve

Probleme, povezane z iskanjem predslik, lahko delimo v nivoje glede na zahtevnost:

1. doloÄŊitev, ali obstajajo predslike za dano trenutno stanje sistema
2. Åątetje predslik za dano trenutno stanje sistema
3. naÅątevanje konfiguracij predslik
4. vpraÅąanje reverzibilnosti sistema
5. jezik vseh stanj GoE

V magistrskem delu opiÅąem algoritem, ki reÅąuje prve tri probleme. Preostala problema sta si sorodna. Problem reverzibilnosti je na sploÅąno dokazano nereÅąljiv [17]. Problem jezika stanj GoE je za 1D preprosto reÅąljiv, za 2D pa je verjetno podobno kakor vpraÅąanje reverzibilnosti nereÅąljiv.

6.1 Analiza 2D CA s pomoÄŊjo konÄŊnih avtomatov

Problem jezika stanj brez predslik bi potreboval teorijo 2D formalnega jezika, ki Åąe ne obstaja. Kljub temu je moÅ¿no zastaviti probleme, ki se jih da preslikati v konÄŊni avtomat. Na primer, ÄŊe obravnavamo vrstice kakor simbole jezika, lahko sestavimo konÄŊni avtomat. Simboli (stanja vrstic) definirajo prehode med stanji konÄŊnega avtomata. Stanja avtomata pa so vsi moÅ¿ni nabori robov vrstice. Pomembna je le prisotnost roba, ne pa tudi uteÅ¿. Poln nabor vsebuje vse moÅ¿ne robove, prazen nabor pa ne vsebuje nobenega roba. Vsako zaporedje vrstic (simbolov), ki v avtomatu pelje iz polnega v prazno stanje, je konfiguracija GoE. Zanke v takem avtomatu pa so sorodne delcem.

Opisani pristop postane nepraktiÄŊen Å¿e za kratke vrstice. Åătevilo vseh moÅ¿nih robov raste eksponentno z dolÅ¿ino vrstice N_x. To Åątevilo, $|S|^{(M_y-1)(M_x-1+N_x)}$, je enako Åątevilu vseh naÄŊinov, na katere se lahko okolici dveh vrstic prekrivata. Stanja avtomata pa so vsi moÅ¿ni nabori robov. Nabor lahko opiÅąemo kakor binarni niz, kjer vsak bit pomeni prisotnost ali odsotnost enega od robov. Vseh stanj konÄŊnega avtomata je tako $2^{|S|^{(M_y-1)(M_x-1+N_x)}}$. Åătevilo stanj avtomata postane neobvladljivo Å¿e za kratke vrstice.

NajkrajÅąa beseda GoE znotraj regularnega jezika stanj GoE je tista, ki po najkrajÅąi poti in brez zank pripelje iz polnega v prazen nabor. Pri 1D CA se je izkazalo, da je dolÅ¿ina najkrajÅąe besede GoE povezana s kompleksnostjo prostorsko-ÄŊasovnih vzorcev, ki se pojavljajo pri danem pravilu. Podobno so verjetno tudi pri 2D CA konfiguracije GoE veÄŊje pri pravilih, ki omogoÄŊajo kompleksno dinamiko delcev. Verjetno bi bilo moÅ¿no to izkoristiti za iskanje zanimivih pravil v okolici quad.

6.2 IzboljÅąave podanega algoritma

Implementacijo algoritma je moÅ¿no izboljÅąati, tako da zahtevnost ne raste tako hitro z velikostjo problema. Za sedaj vidim samo reÅąitev, ki se osredotoÄŊa na povpreÄŊno zahtevnost, namesto na maksimalno zahtevnost. Na primer, sedaj algoritem rezervira pomnilnik za uteÅ¿i vseh moÅ¿nih robov, dovolj pa bi bilo, ÄŊe bi bil rezerviran pomnilnik samo za robove z uteÅ¿jo veÄŊjo od niÄŊ. Sploh pri iskanju stanj GoE je robov malo v primerjavi z vsemi moÅ¿nimi. Tak pristop bi dal dosti manjÅąo porabo pomnilnika in verjetno tudi zniÅ¿al ÄŊas procesiranja.

Morda bi bilo moÅ¿no vnaprej doloÄŊiti zgornjo mejo Åątevila ne-niÄŊelnih robov. To bi omogoÄŊalo zgodnjo oceno, ÄŊe je izraÄŊun za dan problem izvedljiv. Na primer, ÄŊe bi v vrstiÄŊnem procesiranju obravnavali vse zaÄŊetne robove hkrati, namesto vsakega posebej, bi dobili prevelik nabor konÄŊnih robov. Tak nabor bi vedno vseboval vse pravilne konÄŊne robove, in Åąe nekaj napaÄŊnih. Ampak skupno Åątevilo bi bilo lahko obÄŊutno manjÅąe od polnega nabora. To Åątevilo bi lahko uporabili za oceno porabe pomnilnika in ÄŊasa. Na ta naÄŊin bi bilo moÅ¿no reÅąevati probleme, ki so na meji tega, kar zmore sodobna strojna oprema.

Samo implementacijo algoritma je tudi moÅ¿no optimizirati. Univerzalnost algoritma, predvsem univerzalnost velikosti okolice, prispeva dosti reÅ¿ijskih stroÅąkov. ÄŇe bi bil algoritem specializiran samo za quad ali GoL, bi se lahko izognil precej operacijam. Podobno velja za nabor stanj celice. Za binarne CA je moÅ¿no optimizirati operacije, ki pretvarjajo 2D nize celic v cela Åątevila in obratno. Binarne podatke je moÅ¿no kompaktno pakirati v pomnilnik. MogoÄŊe je tudi uporabiti logiÄŊne operacije nad celo besedo celic, namesto procesiranja vsake celice posebej.

Procesiranje v algoritmu poteka v gnezdenih zankah in v znanem vrstnem redu. Le manjÅąi del dostopov do pomnilnika je nakljuÄŊen. Zaporednost podatkov Å¿e sama po sebi omogoÄŊa dobro optimizacijo dostopa do predpomnilnika, tako da na tem podroÄŊju ni dosti oÄŊitnih optimizacij.

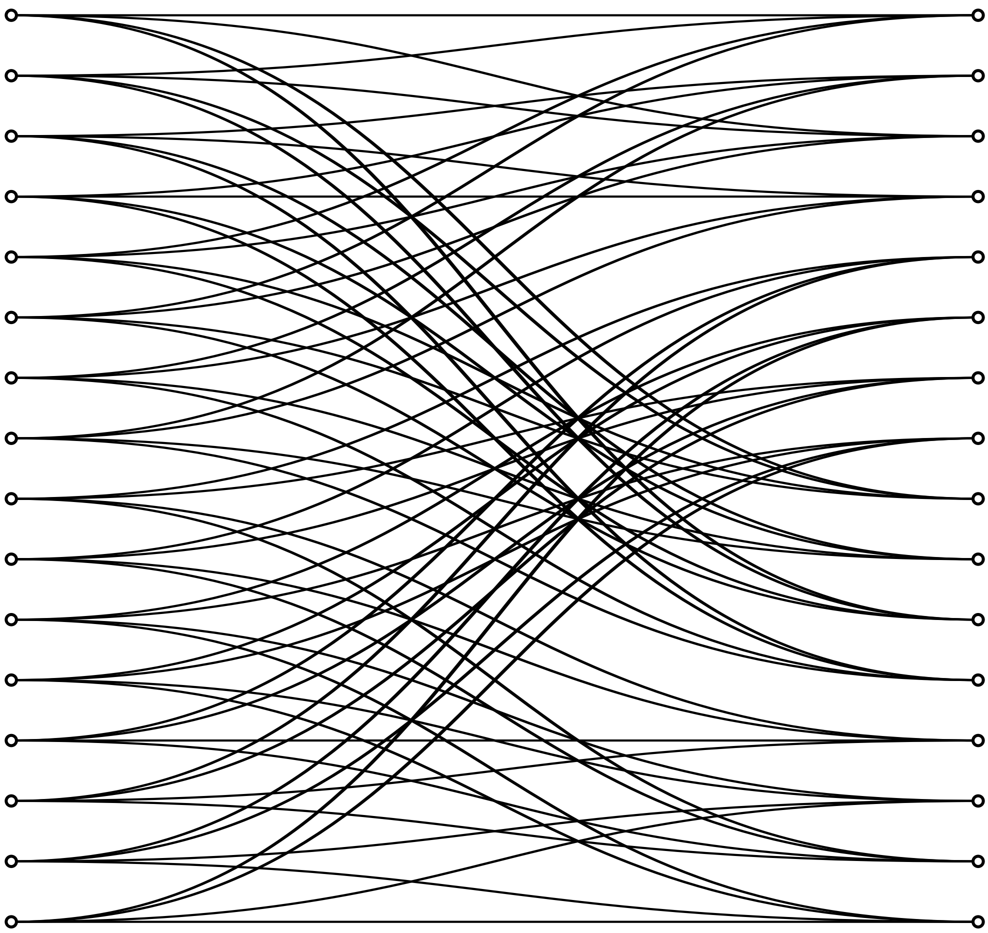
6.3 Dokazovanje

Magistrska naloga je bolj skopa z dokazi. Za 1D CA sem uporabljal neposredno dokaze iz teorije grafov, pri 2D CA pa mreÅ¿a predslik ni obiÄŊajen graf, tako da dokazovanje s teorijo grafov ni tako preprosto. Verjetno pa je moÅ¿no pretvoriti dani graf v dualno obliko s transformacijo zvezda-trikot. ÄŇe obstaja dualna oblika grafa, kjer ni ploskev, temveÄŊ samo vozliÅąÄŊa in povezave, bi jo lahko uporabili za izpeljavo in dokazovanje dejanske raÄŊunske kompleksnosti problema.

Dodatek A

De Bruijnov diagram za GoL

Slika A.1 prikazuje del De Bruijnovega diagrama za GoL. Graf s 16 vozliÅąÄŊi in 64 povezavami je oÄŊitno preveÄŊ kompleksen, da bi bil primeren za razlago algoritma. Åătevilo vseh moÅ¿nih okolic (ploskev v diagramu) pa je celo 512.



Slika A.1: MreÅ¿a ene celice za binarni CA z Moorovo okolico $M_x = M_y = 3$.

Dodatek B

Primer delovanja algoritma za CA z okolico quad

V naslednjem primeru je prikazano Åątetje in iskanje predslik za binarni CA (celica ima lahko 2 stanji) z okolico velikosti 2×2 in pravilom Åątevilka `0x725A`.

Majhna konfiguracija velikosti 2×2 je shranjena v datoteki `test_2x2.cas`. Uporabil bi veÄŊjo konfiguracijo, a na Å¿alost ne poznam pravila, ki bi za veÄŊjo konfiguracijo dal majhno Åątevilo predslik.

Uporabljen je program [12] z verzijo `v1`.

```
$ ./ca2d_preimages 2 2 2 0x725a 2 2 test_2x2.cas
CA parameters:
  sts  : 2
  ngb  : siz={2,2}, a=4, n=16
  ovl-y: siz={1,2}, a=2, n=4
  ovl-x: siz={2,1}, a=2, n=4
  rem-y: siz={1,2}, a=2, n=4
  rem-x: siz={2,1}, a=2, n=4
  ver  : siz={1,1}, a=1, n=2
  shf-y: siz={1,1}, a=1, n=2
  shf-x: siz={1,1}, a=1, n=2

RULE = 0x725a
  tab [0] = [[0,0],[0,0]] = 0
  tab [1] = [[1,0],[0,0]] = 1
  tab [2] = [[0,1],[0,0]] = 0
  tab [3] = [[1,1],[0,0]] = 1
  tab [4] = [[0,0],[1,0]] = 1
  tab [5] = [[1,0],[1,0]] = 0
  tab [6] = [[0,1],[1,0]] = 1
  tab [7] = [[1,1],[1,0]] = 0
```

```
  tab [8] = [[0,0],[0,1]] = 0
  tab [9] = [[1,0],[0,1]] = 1
  tab [A] = [[0,1],[0,1]] = 0
  tab [B] = [[1,1],[0,1]] = 0
  tab [C] = [[0,0],[1,1]] = 1
  tab [D] = [[1,0],[1,1]] = 1
  tab [E] = [[0,1],[1,1]] = 1
  tab [F] = [[1,1],[1,1]] = 0

CA configuration siz [yâĘŞ,xâĘŠ] = [2,2]:
 1 0
 0 1

NETWORK: edge weights from forward/backward direction,
         summed preimages
net [dy=0][y=0] = [1 1 1 1 1 1 1 1 ]
net [dy=0][y=1] = [2 2 0 2 3 3 0 1 ]
net [dy=0][y=2] = [0 3 5 0 0 2 7 2 ]  cnt [0] = 19
net [dy=1][y=0] = [0 4 4 0 0 4 5 2 ]  cnt [1] = 19
net [dy=1][y=1] = [2 0 3 4 2 0 1 1 ]
net [dy=1][y=2] = [1 1 1 1 1 1 1 1 ]

preimage i=0:  CA configuration siz [yâĘŞ,xâĘŠ] = [3,3]:
 1 0 0
 0 0 0
 0 1 0

preimage i=1:  CA configuration siz [yâĘŞ,xâĘŠ] = [3,3]:
 1 0 0
 0 0 0
 0 1 1

preimage i=2:  CA configuration siz [yâĘŞ,xâĘŠ] = [3,3]:
 1 0 0
 0 0 1
 0 1 0

preimage i=3:  CA configuration siz [yâĘŞ,xâĘŠ] = [3,3]:
 1 0 0
 0 0 1
 0 1 1

preimage i=4:  CA configuration siz [yâĘŞ,xâĘŠ] = [3,3]:
 0 1 0
 1 1 0
 1 0 0

preimage i=5:  CA configuration siz [yâĘŞ,xâĘŠ] = [3,3]:
```

```
 0 1 0
 1 1 0
 1 0 1

preimage i=6:  CA configuration siz [yâĘŞ,xâĘŠ] = [3,3]:
 0 1 0
 1 1 0
 0 1 1

preimage i=7:  CA configuration siz [yâĘŞ,xâĘŠ] = [3,3]:
 0 1 0
 1 1 0
 1 1 1

preimage i=8:  CA configuration siz [yâĘŞ,xâĘŠ] = [3,3]:
 1 0 1
 0 0 0
 0 1 0

preimage i=9:  CA configuration siz [yâĘŞ,xâĘŠ] = [3,3]:
 1 0 1
 0 0 0
 0 1 1

preimage i=10:  CA configuration siz [yâĘŞ,xâĘŠ] = [3,3]:
 1 0 1
 0 0 1
 0 1 0

preimage i=11:  CA configuration siz [yâĘŞ,xâĘŠ] = [3,3]:
 1 0 1
 0 0 1
 0 1 1

preimage i=12:  CA configuration siz [yâĘŞ,xâĘŠ] = [3,3]:
 0 1 1
 1 1 0
 1 0 0

preimage i=13:  CA configuration siz [yâĘŞ,xâĘŠ] = [3,3]:
 0 1 1
 1 1 0
 1 0 1

preimage i=14:  CA configuration siz [yâĘŞ,xâĘŠ] = [3,3]:
 0 1 1
 1 1 0
 0 1 1
```

```
preimage i=15:  CA configuration siz [yâĘŞ,xâĘŠ] = [3,3]:
 0 1 1
 1 1 0
 1 1 1

preimage i=16:  CA configuration siz [yâĘŞ,xâĘŠ] = [3,3]:
 0 1 1
 1 1 1
 1 0 0

preimage i=17:  CA configuration siz [yâĘŞ,xâĘŠ] = [3,3]:
 1 1 1
 0 0 1
 0 1 0

preimage i=18:  CA configuration siz [yâĘŞ,xâĘŠ] = [3,3]:
 1 1 1
 0 0 1
 0 1 1
```

Dodatek C

Don Woods opiÅąe svoj algoritem

Prilagam pisno izmenjavo z avtorjem [1] prvega algoritma za iskanje predslik GoL.

```
Don Woods <don@icynic.com>        Thu, Jul 28, 2016 at 11:45 PM
To: iztok.jeras@gmail.com
> Dear Mr. Woods,
>
> I am developing an algorithm for computing preimages of 2D CA. It is based
> on the latest research for computig preimages of 1D CA.
> I am using the 2x2 quad neighborhood in the examples instead of the 3x3
> neighborhood used in Life.
> https://github.com/jeras/fri-magisterij (work in progress and Slovene
> language)
> https://github.com/jeras/preimages-2D (not working yet)
>
> As part of my masters thesis I have to compare my algorithm to existing
> ones.
> You are known as the first person to prove a pattern to be a Garden of Eden
> orphan in the Game of Life.
> Could you send me some references to this proof? I was looking for an
> article to reference, but a generic description (preferably written by you)
> or source code would be as good.
>
> Regards,
> Iztok Jeras
>

Oof, it's been quite a while; let me see what I can remember.

The "proof" was programmatic, i.e. I wrote a program to compute
predecessors to a given position, and the program claimed there were no
predecessors to the particular position that someone had theorised was
```

[1]Don Woods /urlhttp://www.icynic.com/ don/

```
a Garden of Eden.  I did not have a formal proof of correctness for the
program.  (I was a teenager when I wrote it!)  I don't think I still
have the source code anywhere, I'm afraid.  (I might have a paper
printout of it somewhere but don't have time to search right now.)

The program worked by considering each cell in turn and looking at all
possible 3x3 predecessors that would produce the given state.  It then
looked at the next cell to the right, looking for 3x3 predecessors that
were consistent with the 6 overlapping cells from the other, etc.  (In
practice, the 3x3 neighborhoods were indexed so that I could quickly
find which of the 8 possible righthand extensions worked.)  When it had
a 3x(N+2) predecessor for the top row of N, it continued at the left of
the next row, then across that row, etc.  Obviously in those later rows
there was only one new predecessor cell being introduced for each new
current cell after the leftmost, so the combinatoric explosion was not
excessive, particularly since the GoE pattern by design did not have
many potential predecessors even for subsections.  I also deliberately
oriented the target pattern with the short dimension going across, so
that I more quickly reached the point of extending the predecessor one
cell at a time in later rows.

My program normally included a 1-wide border around the area being
backtracked, to ensure there were no artifacts introduced, but for the
GoE proof I restricted it to the non-empty rectangular boundary, which
I think was 9 by 33?  Thus, as I think was mentioned at the time, what
I actually demonstrated was a stronger result: there was no predecessor
that led to a state that included the GoE as a 9x33 subsection.

I hope that helps.  Let me know if you have further questions.

        -- Don Woods

Don Woods <don@icynic.com>        Tue, Aug 16, 2016 at 6:42 PM
To: iztok.jeras@gmail.com
> I found the time to read your response just now.
>
> Your description of the algorithm confirms what I imagined based on third
> party descriptions.
> Your details regarding combinatorial explosion which affects memory
> consumption and processing time were extra helpful.
>
> May I cite this email in my master thesis, since there are no other direct
> sources I would know of?
>
> Regards,
> Iztok Jeras

Permission granted.  Good luck with your thesis!

        -- Don Woods
```

Slike

Literatura

[1] Conway's Reverse Game of Life. https://www.kaggle.com/c/conway-s-reverse-game-of-life, 3 2013.

[2] Neil Bickford. Reversing the Game of Life for Fun and Profit. https://nbickford.wordpress.com/2012/04/15/reversing-the-game-of-life-for-fun-and-profit/, 4 2012.

[3] Peter Borah. Atabot. https://github.com/PeterBorah/atabot, 2013.

[4] Bryan Duxbury. Cellular Chronometer. https://github.com/bryanduxbury/cellular_chronometer/tree/master/rb, 9 2013. ÄŇlanek: https://bryanduxbury.com/2013/09/02/cellular-chronometer-part-1-reversing-the-game-of-life/.

[5] 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.

[6] Dave Greene. New Technology from the Replicator Project. http://b3s23life.blogspot.si/2013/11/new-technology-from-replicator-project.html, 11 2013. Dostop: 2-avg-2016.

[7] Jean Hardouin-Duparc. ÃĂ la recherche du paradis perdu. *Publ. Math. Univ. Bordeaux AnnÃľe*, 4:51–89, 1972-1973.

[8] Jean Hardouin-Duparc. Paradis terrestre dans lâĂŹautomate cellulaire de Conway. *Rev. FranÃğaise Automat. Informat. Recherche Operationnelle Ser. Rouge*, 8:64âĂŞ71, 1974. Dostopno na http://archive.numdam.org/ARCHIVE/ITA/ITA_1974__8_3/ITA_1974__8_3_63_0/ITA_1974__8_3_63_0.pdf.

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

[10] ionreq. calculate previous generations for Conway's Game of Life. https://github.com/ionreq/GoLreverse, 2013. Video prispevek: https://www.youtube.com/watch?v=ZOuMnaarI5k.

[11] 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.

[12] Iztok Jeras. 2D cellular automata preimages count&list algorithm. https://github.com/jeras/preimages-2D, 2016.

[13] Iztok Jeras. Cellular Automata SAGE Toolkit. https://github.com/jeras/cellular-automata-sage-toolkit, 2016.

[14] Iztok Jeras. WebGL QUAD CA sumulator. https://github.com/jeras/webgl-quad-ca, 2016.

[15] 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.

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

[17] Jarkko Kari. Reversibility of 2D cellular automata is undecidable. *Physica D: Nonlinear Phenomena*, 45:379âĂŞ385, 1990.

[18] Paulina A. LĂľon in Genaro J. MartĂŋnez. 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.

[19] E. F. Moore. Machine models of self-reproduction. Objavljeno v *Mathematical Problems in Biological Sciences (Proceedings of Symposia in Applied Mathematics)*, del 14, strani 17–33. American Mathematical Society, 1962.

[20] John Myhill. The converse of Moore's Garden-of-Eden theorem. Objavljeno v *Proceedings of the American Mathematical Society*, del 14, strani 685–686. American Mathematical Society, 1963.

[21] Florian Pigorsch. A SAT-based forward/backwards solver for Conway's "Game of Life". https://github.com/flopp/gol-sat, 10 2015.

[22] Edward Powley. QUAD Prize Submission: Simulating Elementary CAs with Trid CAs. *Journal of Cellular Automata*, 5:415–417, 2010. Dostopno na http://uncomp.uwe.ac.uk/automata2008/files/quadprize_powley.pdf.

[23] Paul Rendell. A Turing Machine In Conway's Game Life. https://www.ics.uci.edu/~welling/teaching/271fall09/Turing-Machine-Life.pdf, 8 2001. Dostop: 2-avg-2016.

[24] Chris Salzberg in Hiroki Sayama. Complex genetic evolution of artificial self-replicators in cellular automata. *Complexity*, 10:33âĂŞ39, 2004. Dostopno na http://www3.interscience.wiley.com/journal/109860047/abstract.

[25] Tommaso Toffoli. Background for the Quad Prize. http://uncomp.uwe.ac.uk/automata2008/files/quad.pdf, 2 2008.

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

[27] Robert Wainwright. Lifeline newsletter - volume 3. http://www.conwaylife.com/w/index.php?title=Lifeline_Volume_3, 9 1971.

[28] Robert Wainwright. Lifeline newsletter - volume 4. http://www.conwaylife.com/w/index.php?title=Lifeline_Volume_4, 12 1971.

[29] Andrew Wuensche in Mike Lesser. *The Global Dynamics of Cellular Automata*. Santa Fe Institute, 1992. Dostopno na http://uncomp.uwe.ac.uk/wuensche/gdca.html.