

Predsluke 2D celičnih avtomatov

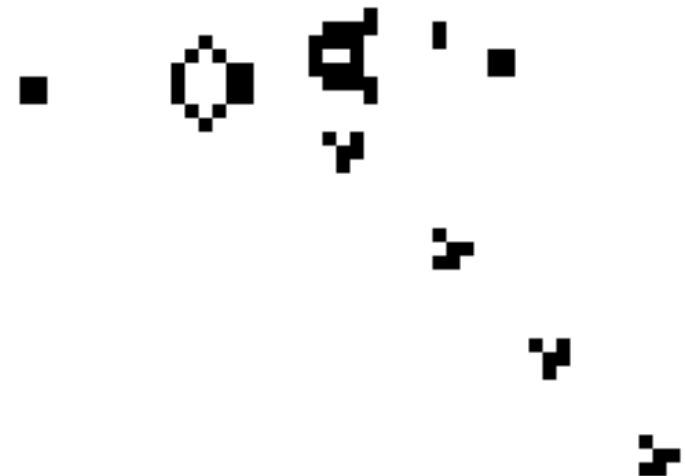
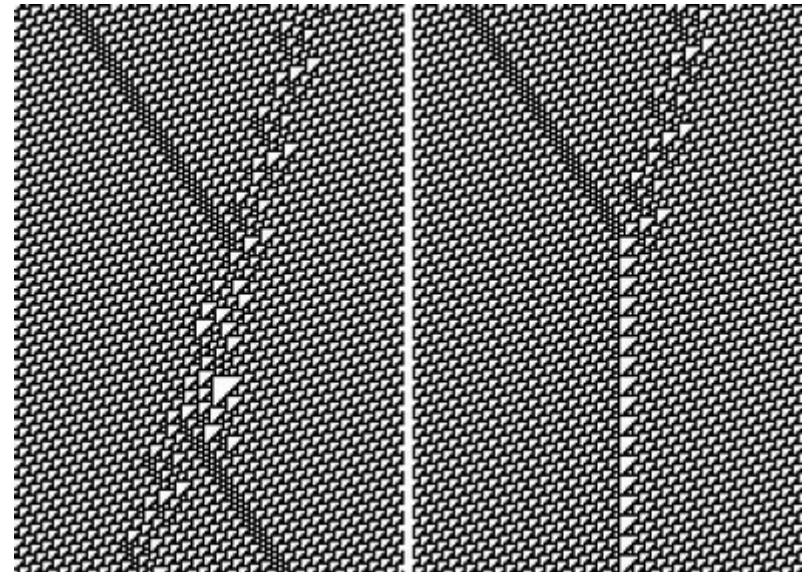
MAGISTRSKO DELO

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Mentor: prof. dr. Branko Šter
Avtor: Izток Jeras

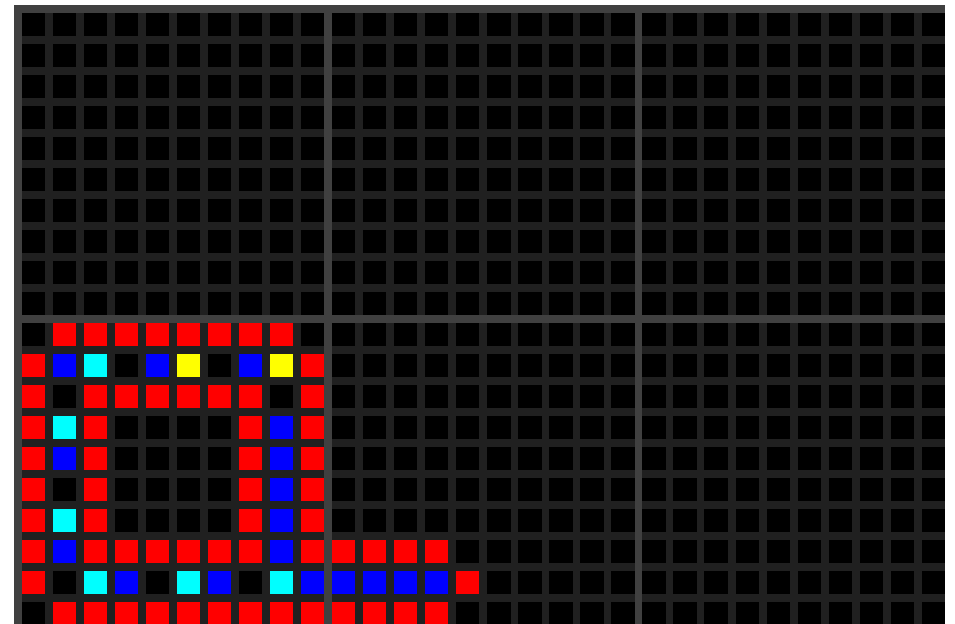
Motivacija: dinamika informacije v ireverzibilnih celičnih avtomatih

- Odnos med izgubo informacije in kompleksnostjo.
- Analiza količine in prostorske razporeditve izgube informacije.
- Pravilo 110:
 - dinamika delcev (trki),
 - stanja Garden of Eden.
- Conwayeva igra življenja:
 - pištola za delce,
 - univerzalni konstruktor,
 - Turingov stroj.



Motivacija: celični avtomati kakor model za probleme v fiziki in biologiji

- Replikacija in evolucija:
 - Langtonova zanka,
 - Evoloop.
- Holografski princip:
 - Termodinamika, entropija in puščica časa.
 - Je vesolje 3D projekcija dogajanja na 2D ploskvi s celicami Planckove velikosti?

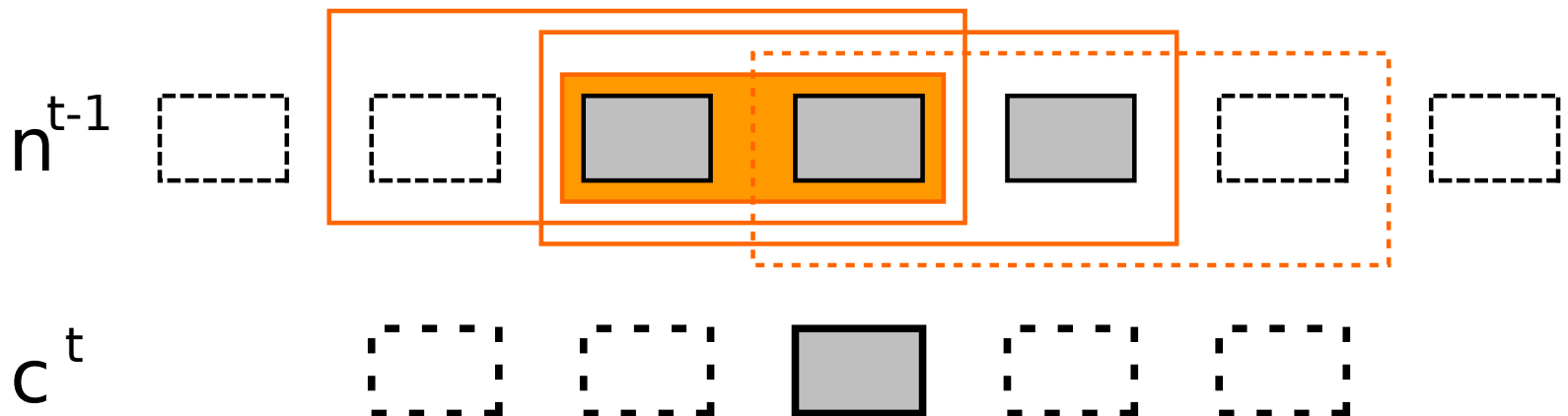


1D CA: pravilo, okolica in prekrivanje

- 1D homogen časovno-prostorsko diskreten dinamični sistem.
- Vrednost celice v sedanosti c_x^t je funkcija vrednosti njene okolice v preteklosti n_x^{t-1} .

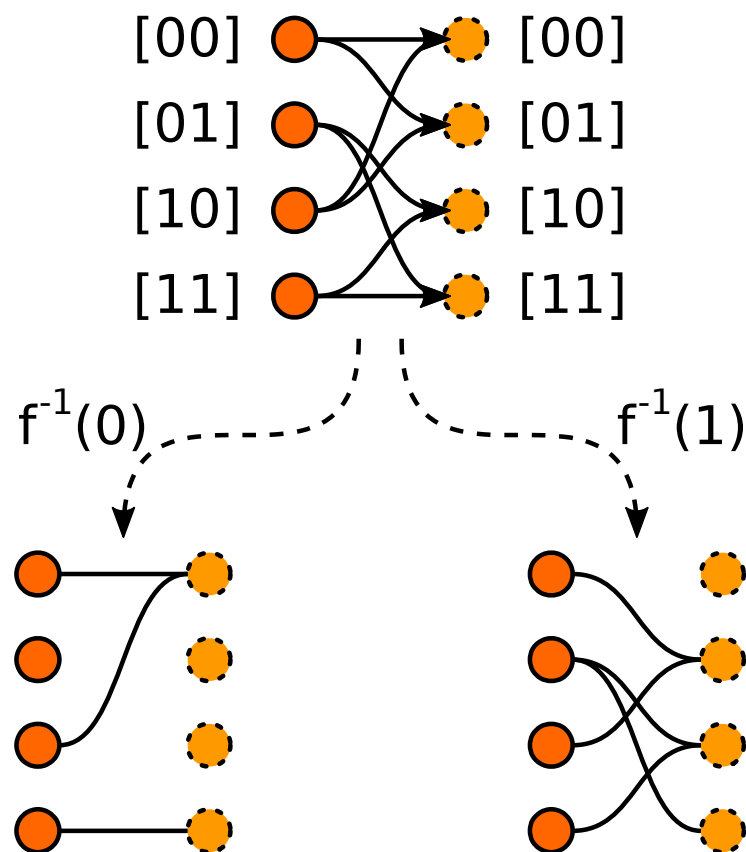
$$c_x^t = f(n_x^{t-1})$$

- Predslike so stanja iz preteklosti, ki se preslikajo v dano sedanost, izpolnjujejo pogoja:
 - vsaka okolica mora izpolnjevati tranzicijsko funkcijo,
 - sosednje okolice se morajo ujemati v prekrivanju (oranžna).



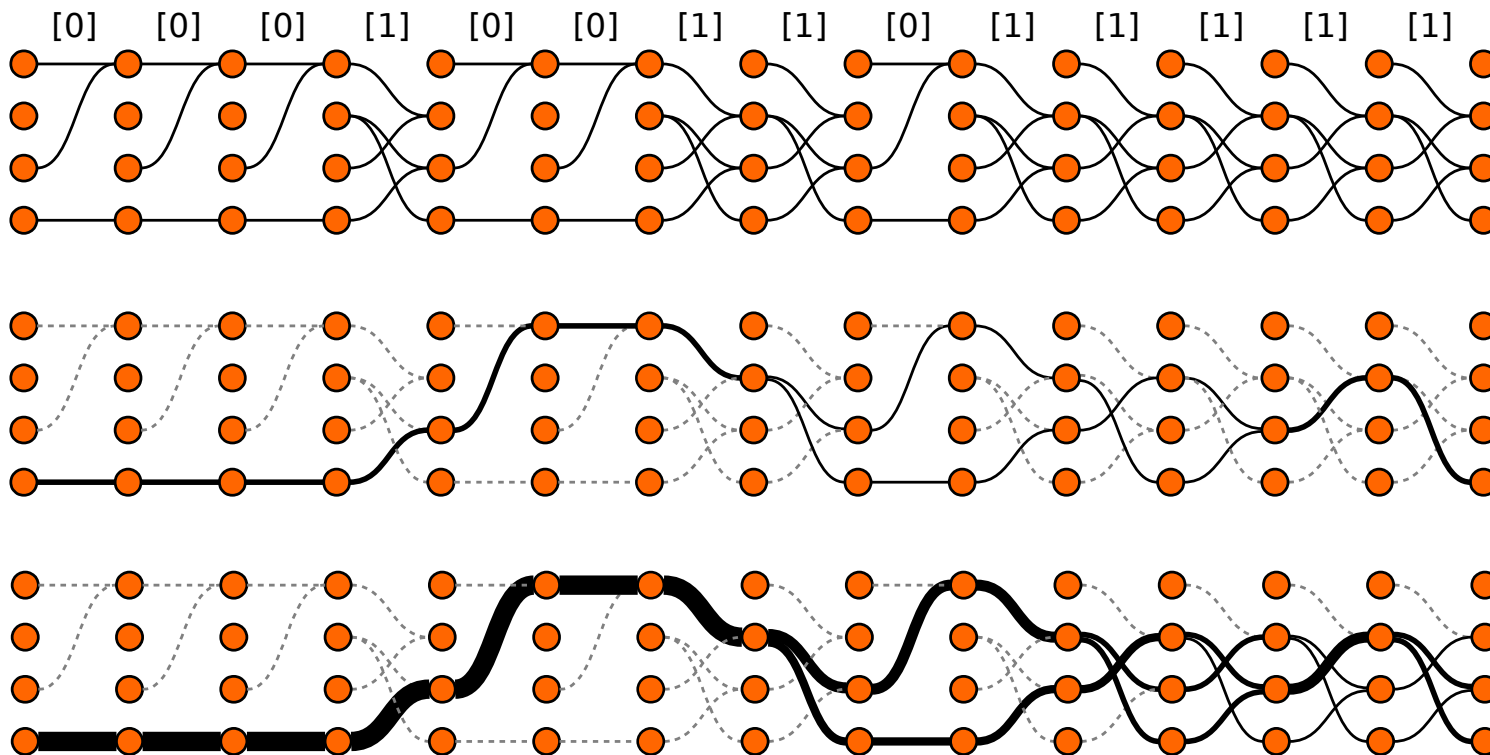
1D CA: graf predslik

- De Bruijnov graf.
- Prekrivanja so volišča (oranžna).
- Okolice so poti med vozlišči.
- Graf se razdeli na grafe za posamezno stanje, vključuje samo okolice, ki pripeljejo v dano stanje.
- Graf predslik predstavlja inverzno tranzicijsko funkcijo $f^{-1}(c)$.
- Primeri kažejo pravilo 110



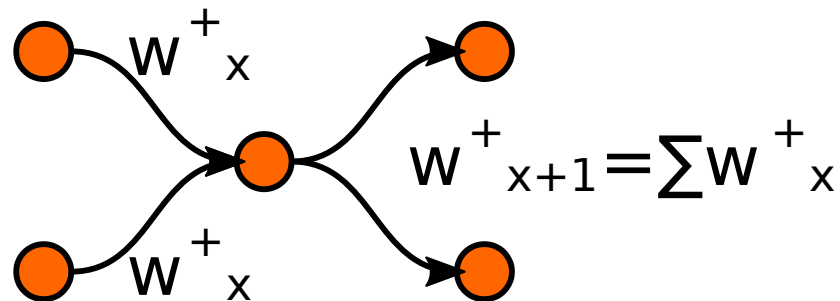
1D CA: mreža predslik

- Z veriženjem De Bruijnovih grafov tvorimo mrežo
- Iščemo vse zvezne poti, ki povezujejo levi in desni rob
- Ostale poti postopno izločujemo

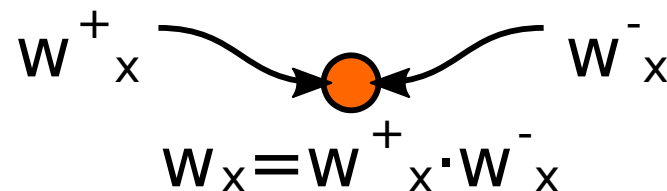


1D CA: uteži v mreži predslik

- V teoriji grafov je utež w enaka številu poti skozi vozlišče.
- Izhodna utež $(x+1)$ je enaka vsoti vhodnih uteži (x) .



- Uteži se računajo v dveh prehodih, naprej (+) in nazaj (-) glede na smer poti v grafu.
- Skupna utež je zmnožek uteži obeh prehodov (+ in -).



1D CA: matrične enačbe

- Za De Brujnov graf vsakega stanja lahko zapišemo z matriko.
- Podobno nizanju grafov, lahko množimo matrike.
- Vsota elementov matrike je število preslik.
- Vsota elementov na diagonali je število preslik za ciklični rob.
- Vmesne rezultate lahko uporabimo za izračun uteži v mreži.

$$D(0) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad D(1) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$D(\alpha) = \sum_{x=0}^{N-1} D(c_x) = D(c_0) D(c_0) \dots D(c_0)$$

1D CA: algoritem za izpis predslik

- Pomnilnik rezerviramo z vnaprej znanim številom predslik (štetje iz desne proti levi).
- Z utežmi za $x=0$ inicializiramo vrednosti.
- Za vsako okolico na položaju x uteži določajo koliko poti nadaljuje z katerim stanjem celice $x+1$.

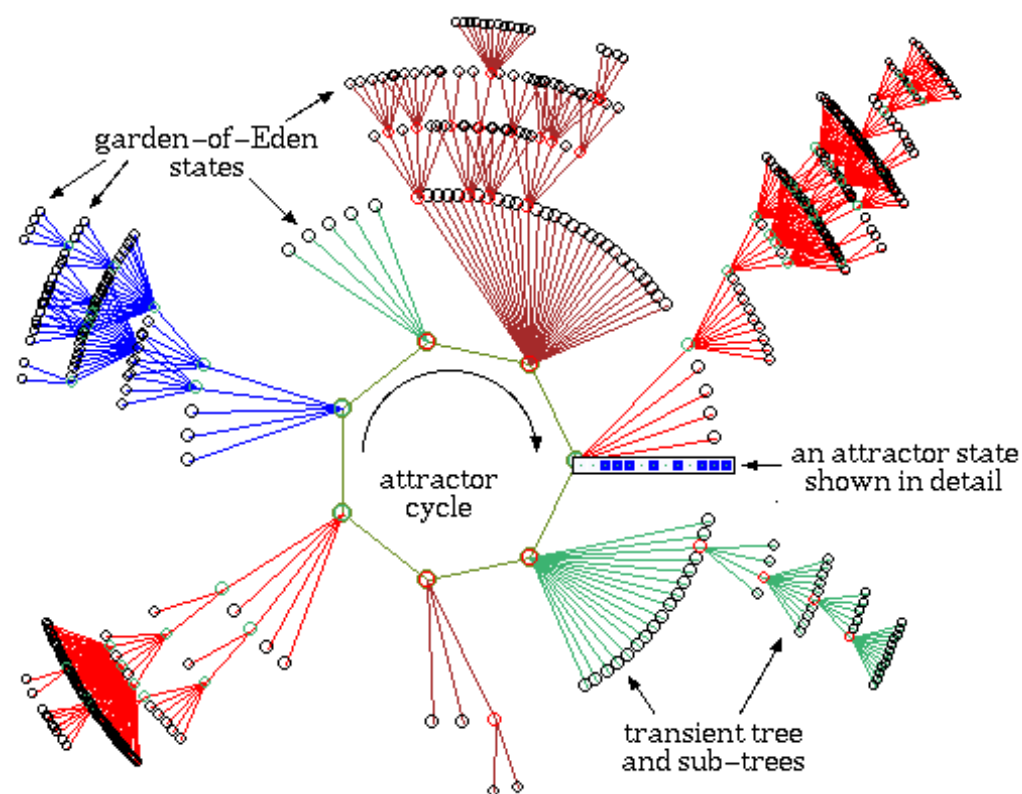
1	1	1	1	1	0	0	0	0	1	0	0	1	0	1	0	1
1	1	1	1	1	0	0	0	0	1	0	0	1	0	1	1	0
1	1	1	1	1	0	0	0	0	1	0	0	1	1	0	1	0
1	1	1	1	1	0	0	0	0	1	0	0	1	1	0	1	1
1	1	1	1	1	0	0	0	0	1	1	1	0	1	0	1	0
1	1	1	1	1	0	0	0	0	1	1	1	0	1	0	1	1
1	1	1	1	1	0	0	0	0	1	1	1	0	1	1	0	1

1D CA: procesna zahtevnost algoritma

- Predvidljiva poraba pomnilnika in časa.
- Štetje predslik (množenje niza matrik):
 - št. operacij $O(N)$: N matričnih množenj,
 - pomnilnik $O(N)$: N vmesnih rezultatov množenja.
- Izpis predslik je odvisno od števila predslik:
 - maksimalna zahtevnost $O(2^N)$,
 - povprečna zahtevnost $O(N)$.

Atraktor in korito

- Prehode med stanji končnega CA lahko sestavimo v usmerjen graf.
- Listi grafa so stanja brez preteklosti GoE.
- V centru grafa je atraktor, ciklično končno stanje v katerega se ustali vsak končni/zaprt dinamični sistem.



1D CA: graf podmnožic in regularni jezik GoE stanj

Primer za pravilo 110:

- Binarna preslikava iz vektorja uteži **$b(x)$** trenutnih vozlišč v naslednji vektor uteži **$b(x+1)$**

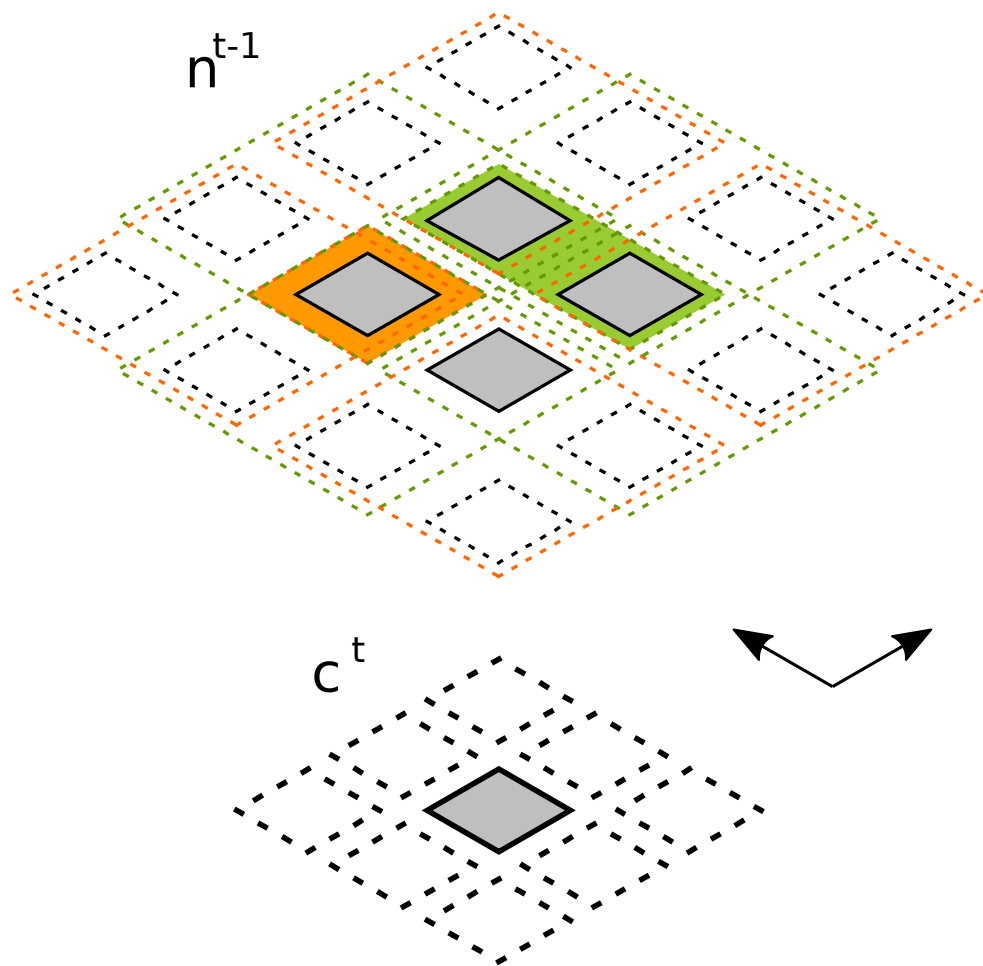
$$b(x+1) = b(x) D(c(x))$$

- Nabor preslikav definira končni avtomat.
- Če je začetno stanje polno in končno stanje prazno, avtomat sprejme GoE besede, in definira regularni jezik GoE stanj.

```
aut = Automaton(  
    (('0000', '0000', 0), ('0000', '0000', 1),  
    ('0001', '0001', 0), ('0001', '0010', 1),  
    ('0010', '1000', 0), ('0010', '0100', 1),  
    ('0011', '1001', 0), ('0011', '0110', 1),  
    ('0100', '0000', 0), ('0100', '0011', 1),  
    ('0101', '0001', 0), ('0101', '0011', 1),  
    ('0110', '1000', 0), ('0110', '0111', 1),  
    ('0111', '1001', 0), ('0111', '0111', 1),  
    ('1000', '1000', 0), ('1000', '0100', 1),  
    ('1001', '1001', 0), ('1001', '0110', 1),  
    ('1010', '1000', 0), ('1010', '0100', 1),  
    ('1011', '1001', 0), ('1011', '0110', 1),  
    ('1100', '1000', 0), ('1100', '0111', 1),  
    ('1101', '1001', 0), ('1101', '0111', 1),  
    ('1110', '1000', 0), ('1110', '0111', 1),  
    ('1111', '1001', 0), ('1111', '0111', 1)),  
    initial_states = ['1111'],  
    final_states   = ['0000'],  
    input_alphabet = [0,1])  
aut_min = aut.minimization()  
list.aut_min.language(5)  
  
[[0, 1, 0, 1, 0]]
```

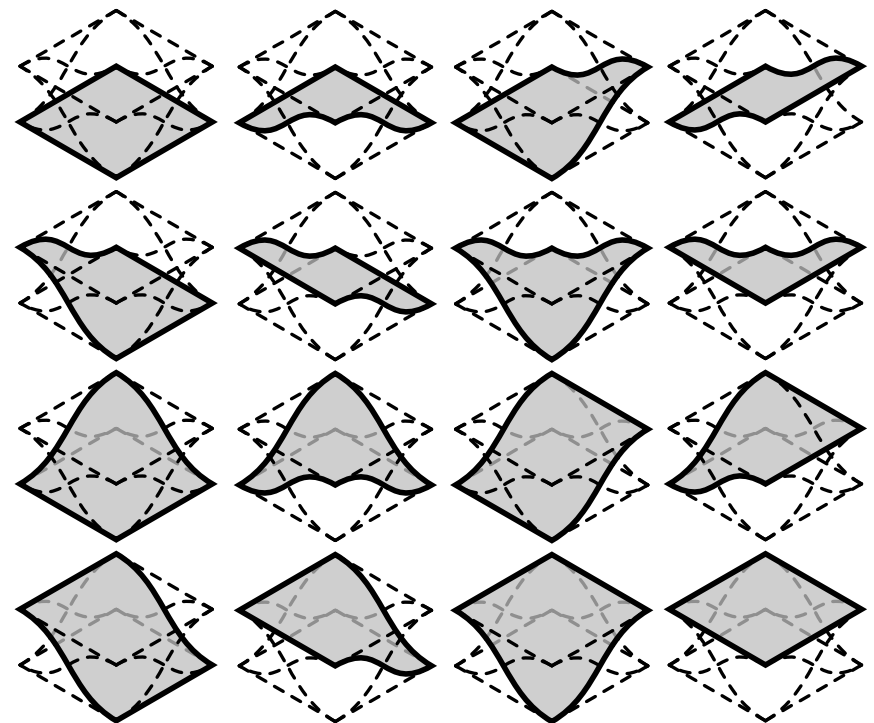
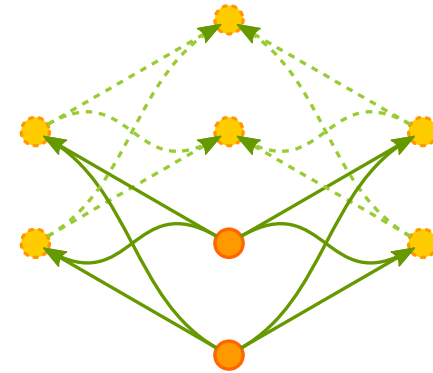
2D CA: okolica in prekrivanje

- Za primer je uporabljena okolica iz 4 celic imenovana **quad**.
- Sosednji okolici v smeri dimenzij X ali Y se prekrivajo za dve celici (**zelena**).
- Sosednji okolice v diagonalni smeri se prekrivajo za eno celico (**oranžna**).
- Za tvorjenje mreže predslik so potrebni drugačni elementi kakor pri 1D problemu.



2D CA: graf predslik

- De Bruijinov graf dobi dodatno dimenzijo v primerjavi z 1D CA.
- Prekrivanje v smeri dimenzij X ali Y je predstavljeno s potmi med vozlišči (**zeleni**).
- Prekrivanje v diagonalni smeri je predstavljeno z vozlišči (**oranžna**).
- Okolice (siva) so predstavljene s ploskvami, katerih vogali so vozlišča in robovi poti med vozlišči, prikazane so vse možne okolice/ploskve.



2D CA: mreža predslik

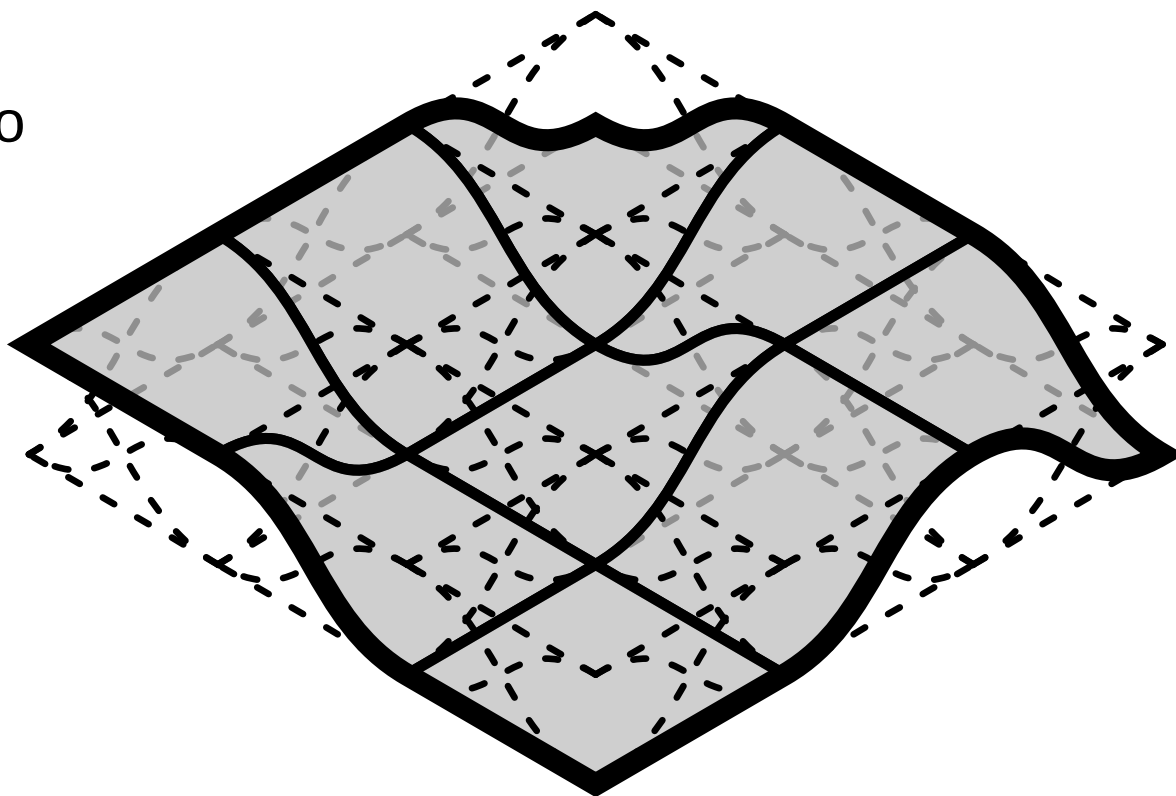
- S tlakovanjem grafov predslik tvorimo mrežo.
- Iščemo vse zvezne ploskve, ki pokrivajo celotno površino mreže.
- Ostale ploskve postopno izločujemo.
- Poudarjena ploskev prikazuje predsliko z vrednostjo:

1110

1001

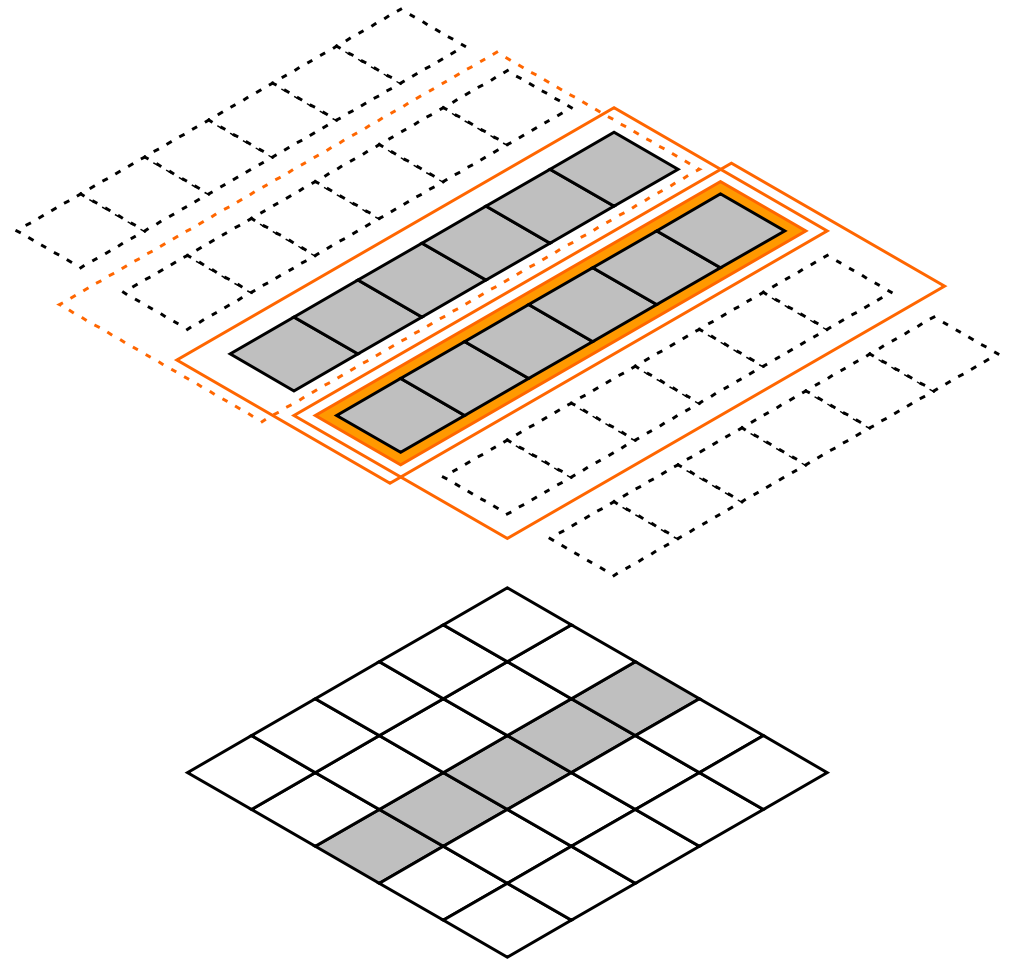
0011

0010



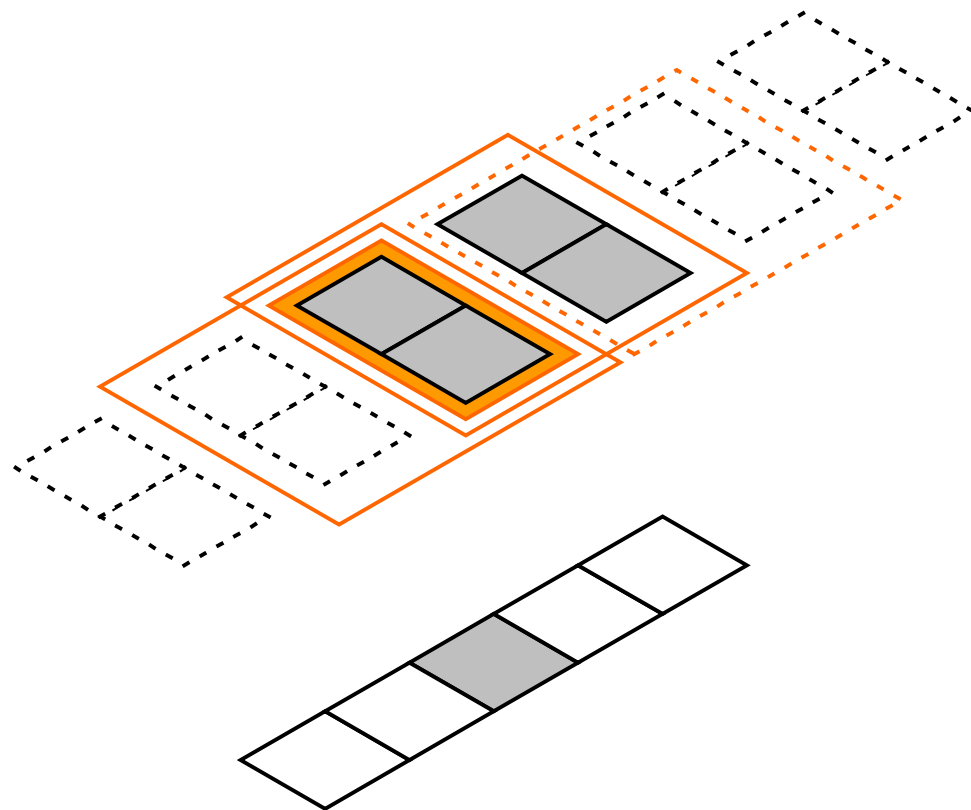
2D CA: pretvorba v 1D problem

- Ploskev razdelimo na vrstice:
 - vsaka vrstica zase je 1D problem, ki ga rešujemo z algoritmom za 1D CA,
 - vrstico obravnavamo kakor kompleksno celico,
 - niz vrstic je spet 1D problem.
- Definirati moramo pretvorbo v graf, opazujemo prekrivanje med predslikami vrstice:
 - vozlišča,
 - povezave?



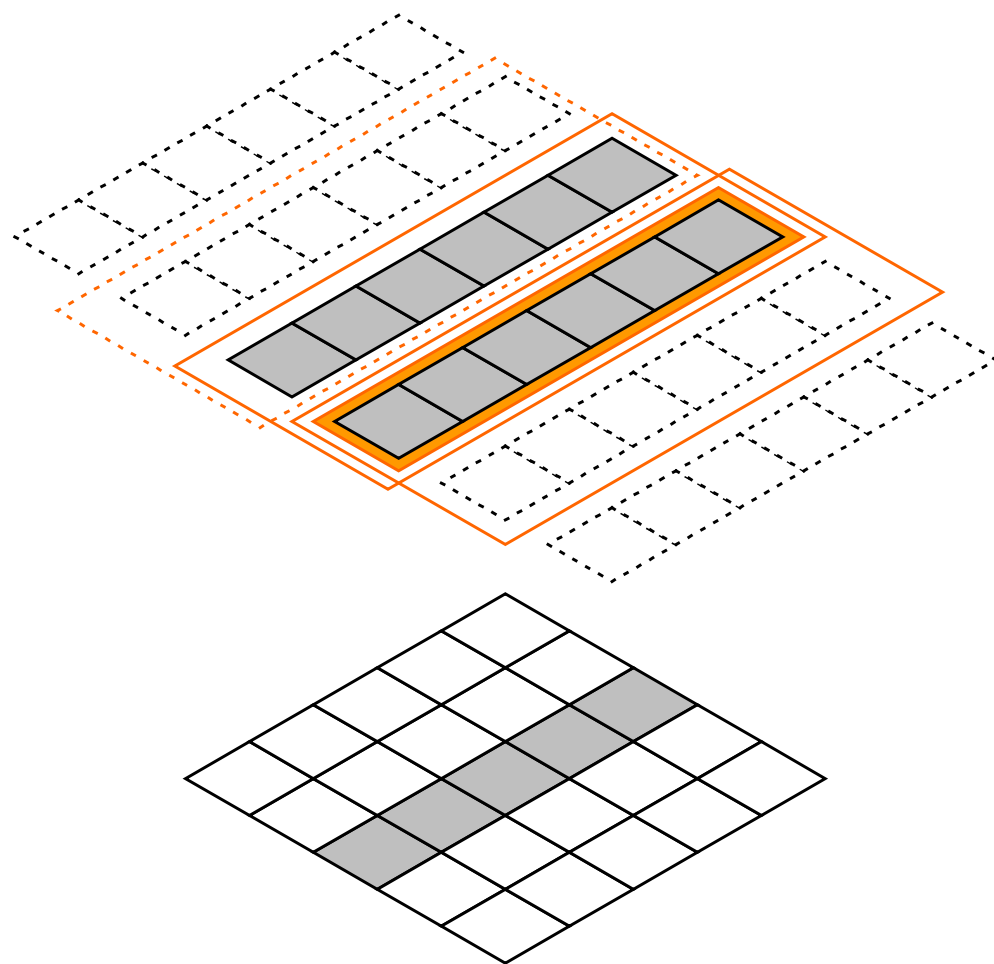
2D CA: štetje predslik znotraj vrstice

- Predslike vrstice so sestavljene iz več vrstic.
- Pretvorba v mrežo predslik:
 - prekrivanja okolic so vozlišča,
 - okolice so poti med vozlišči.
- Algoritem za štetje in izpis predslik se obnaša podobno kakor pri 1D CA.



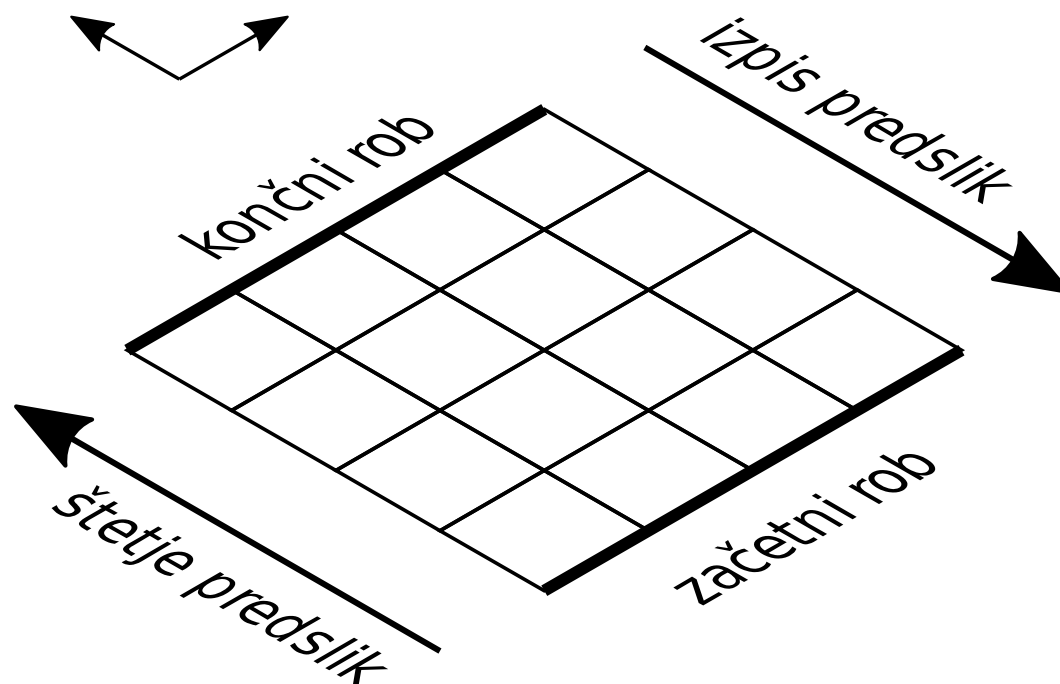
2D CA: štetje predslik po vrsticah

- Vsaka vrstica predstavlja kompleksno celico novega avtomata.
- Pretvorba v mrežo predslik:
 - vsa možna prekrivanja predslik vrstice so vozlišča (**oranžna**),
 - izračunane predslike vrstic so obstoječe povezave.
- Število vozlišč raste eksponentno z dolžino vrstice.



2D CA: izpis predslik

- Smer procesiranja določa, kako so predslike sortirane.



2D CA: stanja Garden of Eden

- Iskanje stanj *Garden of Eden* v celičnem avtomatu *Game of Life* je pogost hobi med raziskovalci CA.
- Algoritem je mogoče poenostaviti tako, da namesto štetja samo preverja obstoj predslik:
 - operacijo *množenja* zamenjamo z logičnim **IN**,
 - operacijo *seštevanja* zamenjamo z logičnim **ALI**.

2D CA: procesna zahtevnost algoritma

- Štetje in izpis predslik vrstice:
 - št. operacij $\mathbf{O}(N_x)$: N_x matričnih množenj,
 - pomnilnik $\mathbf{O}(N_x)$: N_x vmesnih rezultatov množenja.
- Štetje predslik polja:
 - maksimalna zahtevnost $\mathbf{O}(2^{N_x}N_y)$.
- Izpis predslik polja je odvisen od števila predslik:
 - maksimalna zahtevnost $\mathbf{O}(2^{N_x}N_y)$,
 - povprečna zahtevnost $\mathbf{O}(N_xN_y)$.

Rezultati in možne izboljšave

- Alternativa metodam s sestopanjem (Woods).
- Uporaba naprednega algoritma razvitega za 1D CA za predslike 2D CA.
- Ločitev štetja in izpisa predslik.
- Bolj predvidljiva poraba pomnilnika.
- Uporaba razpršenih matrik bi zmanjšala porabo pomnilnika.
- Bolje izbran način kodiranja bi zmanjšal čas, ki se porabi za pretvarjanje iz niza/polja celic v indeks in obratno.

Viri in programska oprema

- Simulator 2D CA z okolico quad:

<https://github.com/jeras/webgl-quad-ca>

- Implementacija opisanega algoritma (C in knjižnica za števila s poljubno natančnostjo GMP):

<https://github.com/jeras/preimages-2D>

- Obstoječ program za izpis predslik GoL:

<https://nbickford.wordpress.com/2012/04/15/reversing-the-game-of-life-for-fun-and-profit/>

- Discrete Dynamics Lab:

<http://www.ddlab.com/>

Vprašanja

izpis predslik

1	1	1	1	1	0	0	0	0	1	0	0	1	0	1	0	1
1	1	1	1	1	0	0	0	0	1	0	0	1	0	1	1	0
1	1	1	1	1	0	0	0	0	1	0	0	1	1	0	1	0
1	1	1	1	1	0	0	0	0	1	0	0	1	1	0	1	1
1	1	1	1	1	0	0	0	0	1	1	1	0	1	0	1	0
1	1	1	1	1	0	0	0	0	1	1	1	0	1	0	1	1
1	1	1	1	1	0	0	0	0	1	1	1	0	1	1	0	1