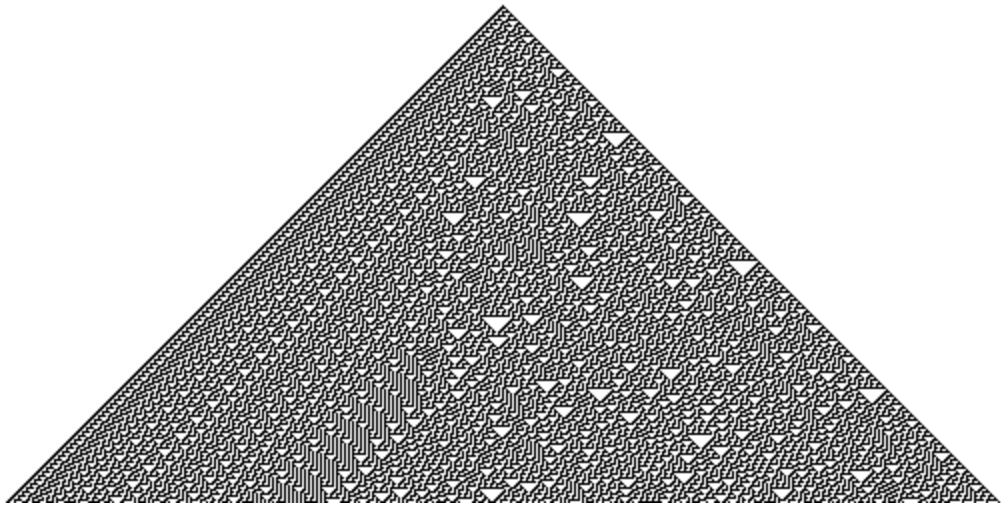
Aihe: Soluautomaattisimulaattori CellSim



Esimerkkikuva miltä simulaation pitäisi näyttää (käytetty 1-ulotteisen automaatin wolfram sääntö nro 30:tä)

Toteutetaan soluautomaattisimulaattori, CellSim.

Käyttäjä: käyttäjä

Toimintoja:

- sisältää 'demoina' wolframin säännöt nro 30 ja nro 110
- simuloi 1-ulotteisia soluautomaatteja eri säännöillä

Muokattavuus:

- mahdollistaa eri sääntömuunnoksien tekemisen/generoimisen
    - naapuruston koko ja muoto valittavissa tietyissä rajoissa
        - naapuruston tilojen määrä pelkästään {0,1} automaatilla 2^k jossa k naapuruston koko
    - solujen tilojen lukumäärä (esim {0,1} tai {0-9})
    - probabilistiset vs deterministiset säännöt
    - solujen synkroninen vs asynkroninen päivittäminen (solut muuttavat tilojansa eri tahtiin)
    - montako kierrosta käydään
    - simulaatiogridin koko valittavissa tietyissä rajoissa (antaa varoituksia jos gridin koko*kierrosten lkm on liian suuri)

Tallennus:

- voi tallentaa automaatin koko simulaation kulun ulkoiseen tiedostoon

Output:

- tuottaa graafisen kuvion automaatin tilojen muutoksista
    - esim 1-ulotteinen automaatti, jossa solujen tilat {0,1} tuottaa kuvion missä 1 tilat ovat mustia pikseleitä ja 0:t valkosia (esim. http://en.wikipedia.org/wiki/File:CA_rule30s.png)

Automaattinen tutkimus:

- analyysimoduuli joka arvioi esim. soluautomaatin kompleksisuutta, muutosten määrää tilojen välillä ym
- generoiva moduuli joka tuottaa erilaisia sääntösettejä annettujen parametrien mukaan
- =>voi yhdessä analyysimoduulin kanssa etsiä potentiaalisesti mielenkiintoisia automaatteja (sanity check: löytää säännöt 30 ja 110)

TODO listaa:

Simulator

- lisää arrayslist tms johon jokainen erillinen vuoro tallennetaan=>tämä voidaan tallentaa, lähettää output luokalle ja analyzer luokalla

UI

- voidaan valita eri optioita, luoda sääntöjä, ajaa simulaatioita jne
- pitää huolen ettei simulaatioon yritetä syöttää virheellisiä parametrejä
- tekstipohjainen (jos tämä sallittua, kysy ohjaajilta)
- tee myös testiluokka jossa kokeillaan eri stringeillä luokan toimivuutta syötteillä
    - tämä samalla testaa koko ohjelman ajamalla tiettyjä konfiguraatioita

RuleSet

- lisää asyncronous vaihtoehto
- lisää rekursiivinen metodi generoimaan naapurustojen permutaatiot
- tee mahdollisesti RuleSet rajapinta, koska asyncroniset ja probabilistiset luokat eroavat sen verran toteutukseltaan ‘normaalista’

Analyzer

- jos aikaa tee eri analyysimetodeja
    - pakkausalgoritmi pakkaa simulaation ja tätä kautta voidaan arvioida miten paljon esiintyy toistoa
    - metodi joka laskee solutilamuutosten määrän vuorojen välillä
- voi konfiguroida generaattoria tulosten perusteleella?

Output

- ottaa vastaan tallennettuja simulaatioita
- tuottaa graafisia ilmentymiä tallenteista

Tallennusominaisuus

- tallentaa simulaation tuotoksen ulkoiseen tiedostoon