Poglavje 1

Amorfno procesiranje

Klasično računalništvo se v današnjem času sooča s tremi temeljnimi problemi. Le ti so povzeti po viru [1] sledeči:

- intenzivnost porajanja napak v zasnovi računalniških sistemov se povečuje z njihovo kompleksnostjo in kompleksnostjo problemov, ki naj bi jih računalniški sistemi reševali; pod pojmom računalniškega sistema smatramo tako strojne, kot tudi programske rešitve;

- arhitektura računalniških sistemov je toga in določena vnaprej, pri čemer se osnovne entitete procesiranja niso zmožne samoorganizirati v domeni nove procesne naloge;

- medprocesne komunikacije delujejo sinhronsko, kar pomeni, da so procesne komunikacije odvisne od takta sistemske ure; možna alternativa je asinhronski koncept procesiranja, kjer se aktivnosti prožijo na osnovi inicializacijskih signalov;

Vsem naštetim problemom se uspešno izogne koncept *amorfnega procesiranja* opisan v pričujočem poglavju.

Za *amorfno* ali *brezoblično* procesiranje (angl. *amorphous processing*) imenujemo kakršnokoli *porazdeljeno* ali *distribuirano* procesiranje velikega števila identičnih paralelno delujočih *entitet* (angl. *particles*) z omejenimi procesnimi zmožnostmi (angl. *limited ability of performance*) [2], [3], [4]. Druga pomembna lastnost amorfnega procesiranja je v lokalnosti interakcij med entitetami, ki se vršijo skozi čas. Soroden primer strukture entitet, ki temelji na omenjenih značilnostih, so celularni avtomati (angl. *cellular automata*). Področje amorfnega procesiranja se začne razvijati sredi devetdesetih let prejšnjega stoletja na uglednem ameriški univerzi MIT [2], izhaja pa iz enovitosti ali homogenosti prostora entitet. Pod pojmom homogenosti imamo v mislih to, da vse procesne entitete izvajajo enak program za interakcijo s svojim sosedstvom.

Amorfno procesiranje omogoča modeliranje po načelih *skupinskega procesiranja*, zato si z njim lahko pomagamo tudi na področju modeliranja fenomenov

iz naravnega okolja kot so dinamike kolonij celic, rojev, jat (angl. *swarms*) in skupinske inteligence (angl. *swarm intelligence*).

Ključni cilj amorfnega procesiranja je doseganje željene dinamike velike množice entitet, ki jo imenujemo za *vzorec*. Področje amorfnega procesiranja naj bi odgovorilo na dve ključni vprašanji, ki se nam pogosto zastavljata pri problemu postavitve strukture večjega števila procesnih entitet. Ti vprašanji sta sledeči:

- Kako zagotoviti *koherentno*[1] ali *soodvisno* željeno dinamiko na osnovi interakcije velikega števila do neke mere nezanesljivih entitet? Pri tem je potebno vzeti v obzir tudi dejstvi, da so entitete lahko povezane nejasno (nepoznavanje neposrednih relacij) in so povezave časovno spremenljive.
- Kakšen lokalni program za interakcijo naj vgradimo v posamezne entitete, da bi dosegli skozi interakcije željeni cilj ali globalni vzorec?

Osnove amorfnega procesiranja predstavimo v sledečih razdelkih.

1.1 Lastnosti amorfnega procesiranja

Osnovne lastnosti amorfnega procesiranja povzete po viru [3] so sledeče:

- število entitet, ki tvori amorfno strukturo na površini ali v prostoru, je izredno veliko (npr. število entitet je velikostnih razredov 10^6-10^{12}) (angl. *masively parallel computer*), vse entitete pa so funkcionalno gledano enake - vsebujejo *enak program*, s pomočjo katerega vršijo svojo interakcijo z okoljem in na osnovi slednje skozi čas menjajo svoja stanja);
- posamezna entiteta je procesno in pomnilniško omejena, lahko zaseda enega od vnaprej definiranih možnih stanj iz končne množice stanj in je sposobna generiranja naključnih števil ali nedeterministične decizije (odločanja);
- entitete vršijo decizijo ali menjajo svoja stanja asinhrono na osnovi svojega lokalnega programa; število entitet ni neposredno odvisno od vsebine lokalnega programa;
- entitete se ne zavedajo svoje absolutne pozicije v prostoru;
- posamezna entiteta svoje stanje spreminja v skladu s svojim programom, pri čemer je spreminjanje stanja pogojeno tudi s stanjem entitet v bližini opazovane entitete, kar poimenujemo za *lokalnost interakcij*; moč vpliva stanja sosedskih entitet na opazovano entiteto pada z njihovo oddaljenostjo; razdaljo vplivnosti med entitetami imenujemo za *komunikacijski radij*;
- globalna dinamika spreminjanja stanj celotne strukture entitet naj bi izkazovala zmožnosti oblikovanje vzorcev, formacijo stabilnih stanj, zmožnosti samoorganizacije in samoreplikacije vzorcev in podvzorcev itd.;

[1] Koherenten - medsebojno povezan, odvisen (Vir: SSKJ).

- posamezne entitete so lahko z vidika zanesljivosti izvajanja programa (spreminjanja lastnih stanj) do neke mere nezanesljive; slednji problem razrešimo z redundanco entitet; na ta način dobimo na globalnem nivoju dinamiko, ki je odporna na napake posameznih entitet (angl. *fault tolerant computing*);

- razmestitev entitet v dvodimenzionalnem ali trodimenzionalnem prostoru je lahko regularna ali neregularna;

- entitete menjajo svoja stanja glede na vplive svojega sosedstva (angl. *peer pressure*), pri čemer je decizijski proces izbire novega stanja posamezne entitete lahko poljuben (lahko temelji npr. na glasovalni tehniki (angl. *voting*), na numeričnih izračunih, na lingvističnih pravilih itd.);

- cilj procesiranja je doseči željeno globalno dinamično časovno obnašanje strukture amorfnih entitet, ki je neodvisno od začetne porazdelitve entitet v prostoru in njihovih začetnih stanj;

- cena posamezne entitete naj bi bila zanemarljiva;

Večino lastnosti amorfnih entitet naštetih v alineah je do neke mere podobnih osnovnim entitetam živih bitij - biološkim celicam. Entitete amorfne strukture komunicirajo preko sporočil, ki se širijo v amorfnem mediju. Moč sporočila praviloma pada s prepotovano razdaljo (npr. s kvadratom razdalje, kot to predvideva Fickov zakon na področjih biologije in kemije).

V zadnjem desetletju raziskave amorfnega procesiranja ne napredujejo več tako hitro zaradi neobstoja cenene procesne platforme, ki bi ob masovnosti uporabe opravičevala tržno ceno rezultata procesiranja. Do ponovnega oživljanja ideje pod imenom *prostorskega procesiranja* (angl. *spatial processing*) pride v zadnjih letih, ko se za potencialno platformo entitete ponujajo *nanocevi* in *biološki procesni mediji*. V slednjem primeru bi lahko za posamezne entitete proglasili celice.

1.2 Izvajanje programa v amorfni strukturi

Množica začetnih stanj vseh entitet v amorfni strukturi naj bi bila ob zagonu enaka [5]. V tem začetnem stanju posamezne entitete pričakujejo sporočila od svojih sosedov, same pa jih ne oddajajo (angl *quescient state*). Brez zunanjega vpliva ali *inicializacije* na posamezne celice tako v takšnem stanju do dinamike ne more priti. Osnovno vodilo inicializacije je čimmanjše število *inicialnih entitet*, kar naj bi zagotavljalo minimalno število posegov iz zunanjega sveta v notranjost amorfne strukture. Stanje strukture v času t si interpretiramo z vektorjem stanj vseh entitet v tej časovni točki. Dinamika je pogojena s programom prehajanja stanj, ki je identičen za vse entitete, ki tvorijo strukturo.

1.3 Razlike med amorfnimi strukturami in strukturami celularnih avtomatov

Področji celularnih avtomatov (CA) in amorfnih struktur (AS) se razlikujeta v naslednjih značilnostih:

- v CA najprej postavimo model strukture (določimo lokalni program in inicializiramo začetna stanja entitet), temu pa sledi simulacija dinamike; z variacijami modela skušamo klasificirati dobljene vzorce dinamike v kontekstu željene evolucije ali zaporedja vzorcev; opisani koncept lahko delno enačimo z analitičnim pristopom, opisanem v začetnem poglavju pričujočega dela;
- na področju AS se v začetku ne usmerimo na definiranje lokalnih interakcij, temveč skušamo na osnovi željene globalne dinamike (vzorca) do lokalnega programa priti na avtomatiziran način;
- osnovne entitete CA so idealno zanesljive, osnovne entitete AS pa nezanesljive pri izvajanju svojega lokalnega programa;
- entitete v AS so lahko v prostoru tudi neenakomerno razporejene, entitete v CA pa zgolj enakomerno;

1.4 Koncepti programiranja amorfnih struktur

Glede na naštete značilnosti entitet amorfmih struktur pri programiranju slednjih lahko uporabljamo naslednje dejavnike:

- *valovna propagacija*: ena od entitet začne v strukturi oddajati sporočilo (angl. *broadcast*); sosednje entitete sporočilo sprejmejo, se vanj „podpišejo“ in ga odpošljejo naprej; entitete na tak način pri sprejemu lahko identificirajo že predhodno sprejetje tega sporočila in njegovo neposredovanje sosedom; na slednji način imamo možnost nadzora nad širjenjem (valovanjem) sporočil in njihovo vzvratno propagacijo;
- *distančna estimacija*: entitete s prejetjem sporočila lahko v domeni lastnega programa (menjavanja stanj) uporabljajo tudi oddaljenost izvira sporočila; slednje je mogoče, če ob zaporedju posredovanja sporočil entitete vpisujejo v sporočila tudi potrdila o sprejemu; slednje omogoča štetje skokov sporočila do opazovane entitete in v domeni radija dosega tudi identifikacijo razdalje izvora sporočila;
- *kontrola regije*: predpostavimo, da imamo dva izvora sporočil (dve izvorni entiteti), ki širita sporočilna vala A in val B; v posamezni entiteti strukture lahko v njenem programu vršimo kontrolo nad širjenjem različnih tipov sporočil (npr. če je preko entitete že odpotovalo sporočilo A, sporočilo B te entitete ne more „prečkati“);

1.5 Specializirani programski jeziki

Ena od temeljnih vej razvoja amorfnega procesiranja je razvoj specializiranih programskih jezikov za opisovanje globalnega vzorca dinamike, ki ga tvorijo s svojo dinamiko posamezne entitete. Programski jeziki naj bi primarno omogočali fokusiranje na željena globalna stanja dinamike strukture, ne pa na samo dinamiko v posameznih entitetah, ki temelji na lokalnih interakcijah. Zgodovinski predhodnik tovrstnih jezikov je bil računalniški jezik Logo.

Osnovni namen amorfnih jezikov je specifikacija globalnih vzorcev dinamike, prevajalniki jezikov pa naj bi v procesu prevajanja zadane časovno spremenljive vzorce prevedli v zakonitosti lokalnih interakcij ali lokalne programe posameznih entitet. Nekaj primerov tovrstnih razvojnih okolij naštejemo v nadaljevanju pričujočega poglavja.

1.5.1 Growing Point Language

Growing Point Language (GPL) je eden od najbolj znanih jezikov za določanje lokalnih interakcij (lokalnega programa entitete) na osnovi globalno definiranih dinamik ali sekvenc željenih vzorcev. Realiziran je kot interpreter. Njegov avtor je Daniel Coore, jezik pa je nastal l.1999 [5]. Temelji na opisovanju željenih vzorcev na osnovi topoloških relacij med točkami (posameznimi entitetami) in objekti (podvzorci sestavljeni iz točk - entitet). Na sliki 1.1 je prikazana tvorba globalnega vzorca, ki ponazarja časovno delovanje inverterja, pridobljena s pomočjo GPL jezika. Slika je povzeta po viru [6].

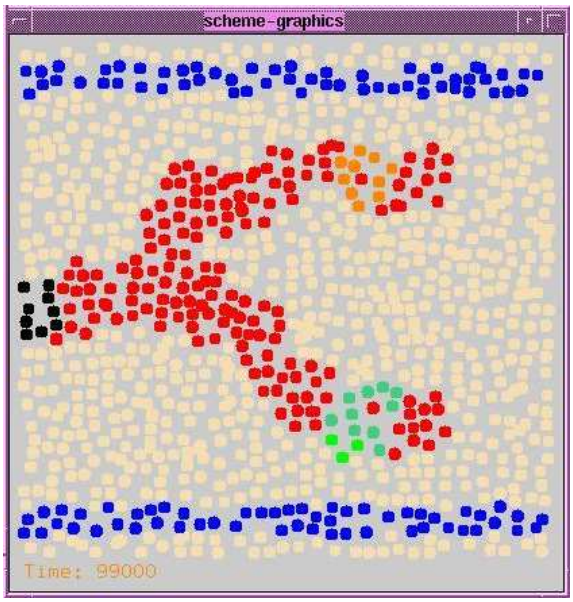
Osnovna vodila zasnove GPL jezika so sledeča:

- uporaba načela *tropizma* (angl.*tropism*): tropizem je biološki fenomen, ki indicira rast ali gibanje kot odziv na okoljske stimulanse; v naravnem okolju se izkazuje kot tendenca naravnih organizmov k črpanju sončne energije (heliotropizem), črpanju kemijskih virov (kemotropizem), izkoriščanja gravitacijskih vplivov (gravitropizem), črpanju vodnih virov (hidrotropizem), črpanju toplotnih virov (termotropizem) itd.;

- uporaba podatkovnega tipa *feromon* (angl. *pheromone*): namenjen je usmerjanju rasti v amorfnih strukturah in je radialno simetričen glede na vir, njegov vpliv pa monotono padajoč z oddaljenostjo od izvira; za dinamiko v amorfnih strukturah je potrebna najmanj ena entiteta, ki vsebuje feromon; če takšne entitete v strukturi ni, dinamika v strukturi ni mogoča;

- vpeljava pojma *rastoče točke*: rastoče točke so prostorsko omejene množice entitet, ki tvorijo željene dinamične podstrukture v kompletni formaciji entitet; v prostoru se lahko kot vzorec premikajo, delijo, spajajo in umirajo; z vidika diskretne časovnosti je rastoča točka venomer prisotna glede na tropizem le v eni od točk, t.i. *aktivni entiteti*; v tej točki lahko rastoča točka odloži material (npr. barvo) ali feromon;

- uporaba podatkovnega tipa *material* (angl. *marker*): podatkovni tip služi kot označevalec, kje je v preteklosti dinamike tekla rast; običajno je im-

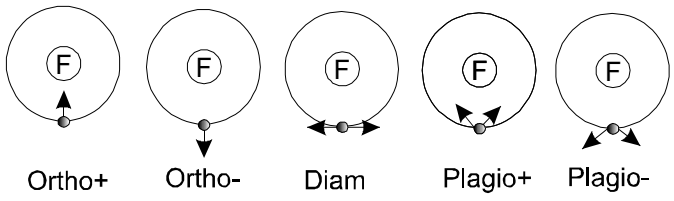


Slika 1.1: Primer vzorca inverterja, pridobljenega s pomočjo GPL jezika [6].

plementiran z barvanjem pozicij entitet; rastoče točke strukture imajo zmožnost zaznave materiala in se glede na njegovo prisotnost odločajo o nadaljnji rasti, lahko pa materiale preoblikujejo (označijo pozicije z novimi materiali);

- vpeljava pojma *poti* rastoče točke: slednja je vidna, če jo rastoča točka ob svojem premikanju označuje z odlaganjem materiala (npr. barvanjem);

Feromoni imajo lahko pozitiven (pospeševalen) ali negativen (zaviralen) vpliv na gibanje rastočih točk. Vrste vplivov feromona na rastočo točko so predstavljene na sliki 1.2, pri čemer puščica na obodu kroga predstavlja vpliv na rastočo točko ob njenem vstopu v vplivno območje feromona.



Slika 1.2: Vrste vplivov feromonov na rast struktur na osnovi zakonitosti tropizma.

Koncept "rastoče točke" predstavlja množico aktivnosti, ki prenaša informacijski pulz preko sosednjih entitet na ciljne pozicije. Pulz se giblje upoštevajoč tropizme, katerih intenzivnost v sosedstvu entitete je spremenljiva. Rastoča točka lahko v entitete prinaša *material* ali *feromone*. Pot rastoče točke je tako množica obiskanih entitet označenih bodisi s tam puščenimi materiali, ali feromoni.

V jeziku GPL i-to rastočo točko definiramo na osnovi njenih *atributov* (`Ai`) in njenih *instrukcij* (`Ii`). Slednje se izvajajo sekvenčno, kot so podane v izvorni kodi programa.

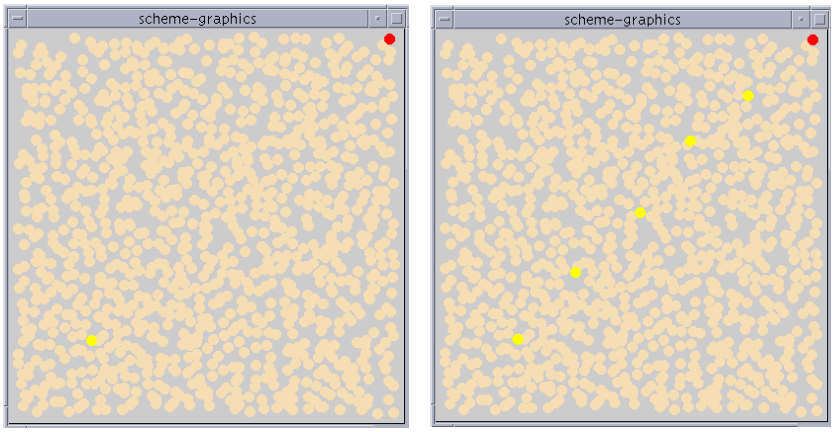
V nadaljevanju si oglejmo primer izvorne kode v jeziku GPL, ki v prostoru entitet oblikuje vzorec povezave (linije) med entitetama A in B. Zgled je povzet po viru [5], njegovo natančnejšo razlago z dodatnimi vzorčnimi primeri pa najdemo v delu [7]. Koda temelji na opisu dveh rastočih točk in sicer statične točke B, ki oddaja privlačni feromon, ter dinamične točke A, ki skuša v obliki rastoče točke ob barvanju, ki poteka v ozadju, doseči pozicijo točke B. Izvorna koda opisa globalnega vzorca je predstavljena v izpisu 1.1.

```
% definicija rastoče točke A
A1:(define-growing-point(A-to-B)
          % A-to-B: ime rastoče točke
A1:    (material A-material)
A1:    (size 0)
A1:    (tropism(ortho+ B-pheromone))
          % tropizme lahko spajamo preko AND, NOT-AND,OR in NOT-OR
    operatorjev
I1:    (for each step
I1:        (when ((sensing? B-material)
I1:            (terminate))
I1:            (default(propagate)))))
% definicija rastoče točke B
A2:define-growing-point(B-point)
A2:    (material B-material)
A2:    (size 0)
I2:    (for-each-step
I2:        (secret+ 10 B-pheromone)))
           %''secret'' ukaz formira tvorbo feromona }
% inicializacija
In:(color
In:    ((B-material) ''red'')
In:    ((A-material) ''yellow'')
In:(with-locations
In:(a b)
In:(at a (start-growing-point A-to-B)
In:(at b (start-growing-point B-point)))
```

Listing 1.1: Izvorna koda povezovalne linije

Na levem delu slike 1.3 povzete po viru [8] je predstavljeno začetno stanje strukture z rumeno obarvano entiteto A in rdeče obarvano entiteto B. Desni del omenjene slike predstavlja propagacijo rastoče točke glede na podani tropizem.

`Material` določa način označevanja dinamike. Če ni specificiran, se pot rastoče točke ne markira. Samo markiranje se izvede po izvedbi vseh preostalih



Slika 1.3: Evolucija linije kot rezultat GPL programa iz izpisa 1.1 [8].

instrukcij. Atribut `size` se uporablja kot obseg markiranja (širina sosedstva, ki ga markiranje zavzame). Če je obseg markiranja po vrednosti 0, se bo markirala le aktivna entiteta. Oglejmo si še opise nekaterih zanimivejših ukazov, ki jih v običajnih programskih jezikih ne najdemo:

- `avoids`: ukaz da feromonu inhibirni (odbojni) vpliv;
- `start-growing-point`: invokacija rastoče točke;
- `propagate`: iskanje nove lokacije rastoče točke (nove lokacije aktivne točke);
- `terminate`: zaključek dinamike rastoče točke;
- `secrete`: definicija obsega vplivnosti feromona;
- `when`: pogojni stavek za vejanje izvajanja akcij;
- `sensing?`: pogojni stavek za detekcijo materiala v aktivni točki;
- `color`: omogoča barvanje materialov;

1.5.2 Origami Shape Language

Koncept jezika Origami Shape Language temelji na konceptu japonskih zgibank iz papirja, kar pomeni, da program za formacijo vzorca zapišemo kot sekvenco pregibov nekega materiala. Z vidika realizacije v tem primeru predpostavljamo, da so entitete opremljene z aktuatorji, ki zgibanje omogočajo. Sekvenco zgibanj, ki jo zapišemo z jezikom, prevajalnik prevede v lokalni program entitet, ki vršijo zgibanje.

Jezik je nastal (l.2001), njegov avtor pa je R. Nagpal. Več o jeziku si lahko bralec prebere v doktorski disertaciji avtorja jezika [9].

1.5.3 Jezik Ecoli

Jezik Ecoli (angl. *extensible calculus of local interactions*) predstavlja nižje nivojski jezik, ki nam omogoča fokusiranje na dinamiko sporočil med posameznimi entitetami in služi kot osnova za določanje programa posamezne celice v navezi z GPL programom. Vsaka GPL sekvenca ukazov se tako najprej prevede v notacijo Ecoli, razvojno okolje slednjega pa omogoča iskanje lokalnega programa.

1.6 Nezanesljivost entitet

Na začetku pričujočega poglavja smo predpostavili, da entitete ne vršijo svojega programa idealno zanesljivo. Zaradi tega bi bilo potrebno pri realnih aplikacijah zagotoviti ustrezno redundanco entitet, ki zagotavlja željeno dinamiko globalnega vzorca. S tem je snovalec razbremenjen skrbi v primeru izpadov posameznih entitet.

1.7 Vztrajnostna jedra

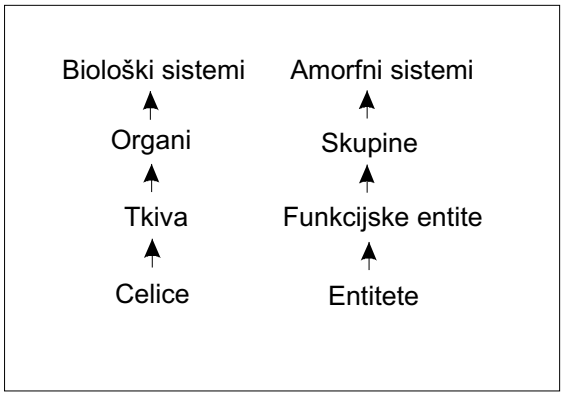
Vztrajnostna jedra (angl. *persistent nodes*) so v amorfnih strukturah strnjene množice entitet v stanjih, ki so z globalnega vidika stabilna. Strnjena množica entitet ima lastnost vztrajnostnega jedra, če ob odvzemu večjega segmenta entitet takšna struktura ima zmožnost svoje obnovitve. Ni nujno, da se dinamika obnovitve izvede v fizično gledano istem sektorju prostora. Lahko se tudi premakne in obnovi nekje drugje v prostoru *amorfnega substrata* (angl. *amorphous substrate*).

1.8 Primerjava amorfnega struktur z biološkimi sistemi

Delovanje amorfnih struktur lahko delno enačimo z delovanjem bioloških sistemov. Omenjena primerjava je prikazana na sliki 1.4, pri čemer pomen ali funkcijo biološke celice povezujemo s pomenom posamezne entitete v amorfni strukturi, pomen biološkega tkiva s pomenom funkcijskih entitet v amorfni strukturi, pomen biološkega organa s pomenom amorfne skupine v amorfni strukturi in pomen biološkega sistema s pomenom amorfnega sistema.

1.9 Povzetek poglavja

Amorfno procesiranje sodi v materialno neodvisne koncepte pristopa k procesiranju (angl. *hardware agnostic*). Z vidika končnih ciljev vodi primarno k aplikacijam *inteligentnih materialov* (npr. *pametnih barv*). Amorfno procesiranje lahko delno povezujemo z danes aktualnim pojmom *vseprisotnega procesiranja* (angl. *ubiquotous computing*).



Slika 1.4: Hierarhija gradnikov v bioloških (levo) in amorfnih sistemih (desno).

Amorfno procesiranje je paradigma, ki neposredno obravnava pojme kot so samoorganizacija, samoreplikacija, samoobnavljanje itd. Glede na koncept sinteze vzorcev je amorfno procesiranje zanimivo tudi iz drugega vidika. Predpostavimo, da v naravi opazujemo proces, pri katerem so razvidni časovni urejeni vzorci dinamike evolucijskega značaja. Ob predpostavki, da nas zanimajo zakonitosti lokalnih interakcij, lahko s pomočjo specializiranih programskih jezikov kot sta GPL in Origami Shape Language pridemo do poznavanja zakonitosti lokalnih pravil.

Literatura

[1] "Manxiu Zhan: Amorphous computing." https://www.nst.ei.tum.de/fileadmin/w00bqs/www/publications/as/2013WS-HS-AmorphousComputing.pdf, January 2014.

[2] "Amorphous computing." http://groups.csail.mit.edu/mac/projects/amorphous/white-paper/amorph-new/amorph-new.html, Maj 2015.

[3] H. Abelson, J. Beal, and G.J.Sussman, "Amorphous computing," tech. rep., 2007. MIT, Technical Report.

[4] H. Abelson, D. Allen, D. Coore, and C. Hanson, "Amorphous computing," *Communications of the ACM*, vol. 94, no. 5, 2000.

[5] "Amorphous computing." http://groups.csail.mit.edu/mac/projects/amorphous/paperlisting.html\#daniel-thesis, Maj 2015.

[6] "GPL inverter." https://groups.csail.mit.edu/mac/projects/amorphous/workshop-sept-99/, November 2016.

[7] "What is language?." http://www.cs.virginia.edu/\symbol{126}evans/cs655-S00/lectures/lecture23.ppt\#380,42,Whatisalanguage?(Lecture1), Maj 2015.

[8] "Programming for swarms." www.cs.virginia.edu/~evans/cs655/projects/zhong.ppt, November 2016.

[9] "Programmable Self-Assembly: Constructing Global Shape using Biologically-inspired Local Interactions and Origami Mathematics." https://pdos.csail.mit.edu/~micahbro/junk/nagpal-thesis%5B2%5D.pdf, December 2017.