Căutând calculatoare în celula biologică
După douăzeci de ani

Discurs de recepție, 24 octombrie 2014
Acad. Gheorghe Păun

Precauții și explicații preliminare

Titlul anterior cere unele precizări pe care mă grăbesc să le aduc încă de la început.

Pe de o parte, el promite prea mult, cel puțin în raport cu preocupările mele din ultimele două decenii și cu discuția care urmează. Este adevărat că există încercări de a pune celula ca atare (bacterii, de exemplu) sau părți ale acesteia (molecule de ADN, în special) să calculeze, dar o direcție de cercetare mult mai realistă, cel puțin deocamdată, și de care m-am ocupat, este cea care caută în celulă *idei* utile informaticii, *modele de calcul,* care, trecând de la structuri și procese biologice la modele matematice, de tip computațional, nu numai că pot asigura o eficiență sporită calculatorului tradițional, electronic, dar se pot și reîntoarce în punctul de pornire, ca instrumente de lucru pentru biologie.

Privind celula prin ochelarii matematicianului–informatician, aceasta este descrierea scurtă și precisă a demersului și aici se plasează experiența personală de cercetare pe care se bazează textul de față.

Pe de altă parte, titlul anunță deja intenția autobiografică. Fiind un moment de sinteză, dacă nu și de bilanț, un discurs de recepție nu poate fi mai puțin autobiografic decât este, se spune, orice roman sau orice volum de versuri. Să nu uităm, viața prin *lumea de curăți și semne* (Ion Barbu) a matematicii impune o mare doză de singurătate, ne-a reamintit acest lucru și acad. Solomon Marcus în discursul său de recepție, iar singurătatea (se presupune că) înțelepțește, dar și îndepărtează de „lumea-ca-lume", de nu mai știi la un moment dat cât dintr-un matematician mai este al „lumii" și cât al matematicii. De aceea, se poate considera că un matematician este autobiografic și în teoremele sale, și în demonstrațiile pe care le găsește teoremelor sale, și în modelele pe care le propune.

Privind retrospectiv, constat că mă aflu la capătul a două perioade de câte două decenii fiecare, ultima dedicată în întregime „căutării de calculatoare în celulă", prima consacrată aproape sistematic pregătirii instrumentelor necesare acestei căutări. Mai ales despre experiența celei de a doua perioade va fi vorba în cele ce urmează.

Un alt titlu posibil

Pentru o vreme, am avut în vedere și un alt titlu, mult mai general, anume, „De la bioinformatică spre infobiologie". Era în același timp o propunere și o previziune, iar paginile care urmează încearcă să dea consistență acestei previziuni. De altfel, ea nu-mi aparține, în mai multe locuri s-a vorbit despre o nouă vârstă a biologiei – la fel s-a prezis și pentru fizică – bazată pe folosirea paradigmelor informațional-computaționale, dacă nu cumva și pe intervenția unor noi capitole de matematică, neelaborate încă. Este vorba nu despre aplicarea informaticii, fie ea teoretică sau practică, în biologie, ci de trecerea la un alt nivel, la o abordare sistematică a fenomenelor biologice în termeni de inspirație computațională, preponderența informației fiind implicită. Încercări care ilustrează această posibilitate, chiar necesitatea ei, pot fi găsite în multe locuri, mergând înapoi în timp până la E. Schrödinger și John von Neumann. Într-o carte recentă, *Infobiotics.*

Information in Biotic Systems (Springer-Verlag, 2013), Vincenzo Manca pledează şi el pentru „o nouă biologie", pe care o numeşte *infobiotică*, plecând de la observaţia că *viaţa este prea importantă pentru a fi investigată numai de biologi.* Aş reformula: *viaţa este prea importantă şi prea complexă pentru a fi investigată numai de biologia tradiţională* – cu sublinierea că, de fapt, tocmai biologii sunt cei chemaţi, nu numai să fie beneficiarii, ci şi să dea contur infobiologiei. Alături de informaticieni şi, cu mult mai plauzibil şi mai eficient, preluând de la aceştia idei, modele, tehnici, apropriindu-şi-le şi dezvoltându-le. Este aici o pledoarie pentru multi-inter-trans-disciplinaritate (începând cu studiile universitare), dar şi un semnal: nu numai că este nevoie, dar, se pare, este şi timpul.

Contextul

Cu titlul de acum în faţă şi căutându-i un gen proxim „oficial", prima sintagmă care ni se oferă este „calculul natural" (*natural computing*) – cu menţiunea, însă, că ea acoperă o foarte largă varietate de cercetări, incluzând bioinformatica şi tinzând şi spre infobiologie. Pentru a avea o descriere autorizată, să mergem la *Handbook of Natural Computing*, editat de Grzegorz Rozenberg, Thomas Bäck şi Joost N. Kok, în patru volume masive, la Springer-Verlag, în 2012. De la începutul prefeţei, aflăm că „acesta este domeniul de cercetare care investighează modele şi tehnici computaţionale inspirate din natură, dar investighează, în termeni de procesare a informaţiei, şi fenomene care au loc în natură". Generalitatea este evidentă, adăugând la dorinţa de a identifica în natură (atenţie, nu numai în biologie) idei utile informaticii, un punct de vedere care, spuneam şi mai devreme, chiar dacă nu este complet nou, pus sistematic în funcţiune poate duce la o nouă paradigmă în cercetarea biologică şi nu numai: abordarea în termeni de procesare a informaţiei, depăşind deci abordarea tradiţională, de tip chimico-fizic.

Ideea a fost formulată şi în alte contexte: punctul de vedere computational (la procesarea de informaţie se adaugă aspectul esenţial al calculabilităţii) poate conduce la o nouă fizică – este, printre alţii, previziunea lui Jozef Gruska, mare promotor al calculabilităţii cuantice, un *pionier al informaticii*, răsplătit ca atare cu o diplomă de către Secţiunea de Informatică a IEEE (să amintim că şi Grigore C. Moisil a primit o asemenea diplomă). Pe aceeaşi teză este construit şi volumul colectiv *A Computable Universe. Understanding and Exploring Nature as Computation,* editat de Hector Zenil şi publicat de World Scientific în 2013, cu multe capitole incitant-entuziaste („Ipoteza universului calculabil", „Universul ca un calculator cuantic") şi cu o prefaţă de lungimea unui capitol, semnată de Roger Penrose, nu totdeauna de acord cu ipotezele din carte.

De altfel, handbook-ul de calcul natural invocat mai devreme include calculul cuantic printre domeniile pe care le are în vedere. Iată cuprinsul lui (secţiunile mari, fără capitole): Automate celulare, Calcul neural, Calcul evolutiv, Calcul molecular, Calcul cuantic, Algoritmi inspiraţi din natură, Modele alternative de calcul. Există o doză de „anexionism" aici (de pildă, automatele celulare nu prea au de a face cu celulele din biologie), dar să reţinem că secţiunea dedicată calculului molecular acoperă *calculul cu ADN, calculul membranar* şi *procesarea genelor la ciliate*, primele două fiind exact ceea ce ne preocupă în continuare.

Popularitatea unui domeniu

Rămânând numai la nivel editorial şi al conferinţelor (nu mai adaug şi proiectele de cercetare, finanţarea, deci), se poate constata că există o veritabilă modă a calculului natural, mai general, a calculabilităţii neconvenţionale, mai restrictiv, a bioinformaticii.

Doar câteva exemplificări: Editura Springer are o serie separată dedicată monografiilor de calcul natural, chiar aşa intitulată, ba chiar şi o revistă – *Natural Computing*. Există o conferinţă *Unconventional Computing,* devenită între timp, uşor pleonastic, *International Conference on Unconventional Computation and Natural Computation.* BIC-TA, adică *Bio-Inspired Computing – Theory and Applications,* este o altă conferinţă de real succes, judecând după numărul de participanţi, al cărui format l-am stabilit, împreună cu colegi din China, în 2005, şi care se organizează de atunci anual în Extremul Orient, ceea ce explică participarea masivă, cercetătorii chinezi fiind foarte activi în acest domeniu.

Am ajuns astfel la genul proxim cel mai restrictiv: calculabilitate inspirată din biologie. Este important de notat că termenul „bioinformatică" are un dublu înţeles, cu determinare, ca să spunem aşa, geografică. În „Vestul pragmatic", el acoperă mai ales aplicaţiile informaticii în biologie (în scenariul „standard", se pleacă de la probleme spre instrumente, fără a teoretiza prea mult). În Europa, ambele direcţii de influenţare sunt avute în vedere – dinspre biologie spre informatică şi invers. Deşi este natural ca aceste direcţii să se dezvolte în paralel, în colaborare, lucrurile nu stau totdeauna aşa. În căutare de soluţii pentru probleme curente, unele realmente urgente, din domeniul biomedical, de exemplu, matematica şi informatica vin adesea cu instrumente puse la punct în alte domenii. Exemplul tipic sunt ecuaţiile diferenţiale, cu o glorioasă istorie în fizică, astronomie, mecanică, meteorologie şi care sunt „împrumutate" de biologie, nu totdeauna cu verificarea adecvării lor. Voi reveni asupra acestui subiect, de mare importanţă pentru promovarea unor instrumente noi.

„Strategia europeană", a construirii unei teorii matematice, care abia ulterior îşi găseşte aplicaţii, are atractivitatea şi avantajele ei – dar şi capcanele ei. European fiind, matematician fiind, am fost mult mai atras de această strategie, cu vremea devenind însă tot mai interesat de „realitate", de aplicaţii.

Ce înseamnă a calcula?

Revin la titlu, cu întrebarea fundamentală privind definiţia noţiunilor de calcul şi de calculator. O întrebare de acelaşi gen cu „Ce este matematica?", având multe şi variate răspunsuri, niciunul satisfăcător pe deplin. Dacă procesarea de informaţie este calcul, atunci putem vedea calcule peste tot în jur. Cu un detaliu foarte important ascuns în formularea dinainte: *putem vedea*, noi, oamenii. Un observator adică, pentru care un proces are semnificaţia unui calcul. Nu vreau să împing discuţia până la speculaţii de genul „căderea unui copac, într-un lac, la mijlocul unei păduri pustii, face sau nu zgomot, ţinând seama că nimeni nu-l aude?" – subliniez însă că întrebarea a făcut obiectul unei lucrări acceptate acum câţiva ani la o conferinţă de calcul neconvenţional – şi nici nu vreau să-L consider pe Dumnezeu, omniprezentul, omniscientul, observator universal (decât, cel mult, pentru zgomotele din pădurile fără alţi martori, nu pentru calcule...).

Un exemplu la limită, dar sugestiv, este cel al unei picături de lichid care cade prin aer. În vremea asta, ea „rezolvă" instantaneu prin forma pe care o ia, o ecuaţie diferenţială. Este acesta un calcul? N-aş merge atât de departe. La fel cu tot ce se întâmplă în celulele unei frunze sau din corpul unui om, la nivel biochimic sau chiar la nivel informaţional.

Ideea calculului ca proces considerat ca atare de un observator nu este nouă. Una dintre concluziile cărţii lui John Searle *The Rediscovery of the Mind* (MIT Press, 1992) este tocmai aceasta, că un calcul nu este o proprietate intrinsecă a unui proces, ci este „observer-relative".

O formulare sugestivă a necesității observatorului în definirea unui proces ca fiind un calcul îi aparține lui Tommaso Toffoli. Citatul care urmează apare într-un articol cu un titlu-afirmație: „Nothing makes sense in computing except in the light of evolution" (*Journal of Unconventional Computing*, vol. 1, 2005, pp. 3-29).

„Tocmai am văzut că nu este util să numim *calcul* orice cuplare netrivială între variabile de stare. Dorim în plus ca această cuplare să fie *intenționat* stabilită pentru a prezice sau manipula – cu alte cuvinte, pentru a *cunoaşte* sau a *face* ceva. (...) Dar acum apar alte întrebări, cum ar fi: *Stabilită de către cine sau ce? Cu ce scop? Cum putem recunoaşte intenționalitatea?*

Departe de mine gândul de a strecura considerații animiste, spiritualiste sau măcar antropice în definiția calculului! *Conceptul de calcul trebuie să apară ca o construcție naturală, bine caracterizată, obiectivă, care să fie recunoscută ca atare şi utilă oamenilor, marțienilor şi roboților deopotrivă*." (subl. n., Gh.P.)

Întrebările lui Toffoli sunt de ținut minte şi de discutat, dar ne îndepărtează de subiect. Să revenim la John Searle şi la o lectură tehnică a ideii de calcul care implică un observator, întreprinsă de Matteo Cavaliere şi Peter Leupold, amândoi fiindu-mi studenți la şcoala de doctorat de la Tarragona, Spania, cel dintâi unul dintre primii mei doctoranzi de acolo. Ei au o serie de lucrări cu acest subiect, citez doar una recentă a lui Peter Leupold, „Is computation observer-relative?", prezentată la *Sixth Workshop on Non-Classical Models of Automata and Applications*, Kassel, Germania, iulie 2014. De fapt, în abordarea celor doi apar, implicit, doi observatori, unul – să-l numim observator de ordinul întâi – care urmăreşte un proces simplu, „traducând" într-un limbaj exterior paşii efectuați de proces, iar al doilea, mai aproape de observatorul lui Toffoli, interpretând drept calcul rezultatul activității primului observator. Cavaliere şi Leupold consideră o serie de perechi proces-observator de ordinul întâi care, separat, au o putere (de calcul) redusă, dar care împreună conduc, din punctul de vedere al observatorului de ordinul al doilea, cel exterior, la puterea de calcul a maşinii Turing.

Maşina Turing

Să pornim însă din altă direcție, de la accepțiunea dată de matematică termenului de calcul. Deja de prin anii '30 ai secolului trecut avem o definiție a ceea ce este calculabil, răspunsul pe care Alan Turing l-a dat întrebării „ce se poate calcula mecanic?", formulate în 1900 de David Hilbert. „Mecanic", deci „algoritmic", reformulăm noi astăzi. Au existat mai multe propuneri (amintesc doar funcțiile recursive şi lambda-calculul), din partea unor nume mari ale matematicii-informaticii (îi amintesc doar pe Alonzo Church, Stephen Kleene, Emil Post), dar soluția pe care Turing a dat-o aşa-numitei *Entscheidungsproblem* a lui Hilbert, ceea ce el a numit *a-maşină* iar astăzi este cunoscut ca *maşină Turing,* a fost acceptată ca fiind cea mai convingătoare (fapt atestat până şi de exigentul Gödel). Acesta este în informatică modelul standard de algoritm (nu am spus *definiție*, pentru că nu avem decât un înțeles intuitiv al ideii de *algoritm*, dar putem spune că pe această cale dispunem de o definiție a ceea ce este *calculabil*).

Fără a intra în amănunte, menționez că problema lui Hilbert, a zecea în lista sa, era mai generală. Ea pleca de la rezolvarea algoritmică a ecuațiilor diofantice (cele cu coeficienți numere întregi), dar în formulare Hilbert spunea: „*Entscheidungsproblem* [problema deciziei în logica de ordinul întâi] va fi rezolvată atunci când vom avea o procedură care să ne permită ca pentru orice expresie logică să decidem printr-un număr finit de operații dacă ea este validă sau satisfiabilă... *Entscheidungsproblem* trebuie considerată ca principala problemă a logicii matematice." La acest nivel general,

teoremele lui Gödel răspund negativ programului lui Hilbert, iar problema a zecea a fost rezolvată complet – tot negativ – în 1970, de Yurii Matijasevich (după eforturi îndelungate ale mai multor matematicieni: Julia Robinson, Hilary Putnam, Martin Davis). Şi Turing aduce un rezultat de tip negativ, el nu numai că a definit „frontierele calculabilului", dar a şi produs un prim exemplu de problemă plasată dincolo de aceste frontiere, nerezolvabilă algoritmic, *problema opririi* (nu există un algoritm – deci, o maşină Turing – care, dându-i-se o maşină Turing arbitrară, cu date iniţiale arbitrare, să spună dacă maşina se opreşte vreodată sau calculează neîncetat). La problema opririi, de altfel, se reduc, direct sau indirect, toate demonstraţiile ulterioare de nedecidabilitate.

Importanţa maşinii Turing pentru informatică, inclusiv pentru calculul natural, este atât de mare, că merită să zăbovim puţin asupra ei.

Câteva detalii mai tehnice

Este interesant să notăm că atunci când şi-a definit „maşina", Turing a plecat explicit, o spune încă de la începutul articolului, de la încercarea de a abstractiza modul în care calculează un om, reducând la minimum resursele şi operaţiile efectuate de acesta. S-a ajuns astfel la un „calculator" care constă dintr-o *bandă* potenţial infinită, mărginită la stânga, împărţită în *celule*, în care se pot scrie *simboluri* dintr-un *alfabet* finit, dat; aceste simboluri sunt citite şi rescrise de un *cap de citire-scriere*, care citeşte o singură celulă, o poate modifica, apoi poate rămâne pe loc sau se poate muta la celula din stânga sau la cea din dreapta; activitatea capului de citire-scriere este controlată de o *memorie*, care se poate afla, la rândul ei, într-un număr finit de *stări*. Avem astfel de a face cu *instrucţiuni* de forma $s_1 a \rightarrow s_2 b\ D$, cu semnificaţia: în starea s_1, citim simbolul a, trecem în starea s_2, modificăm a în b (a şi b pot fi egali) şi deplasăm capul de citire-scriere aşa cum ne indică D. Se porneşte cu banda goală, cu maşina aflată într-o stare s_0 fixată, numită *stare iniţială;* se scriu pe bandă datele iniţiale (de exemplu, două numere care trebuie înmulţite), se poziţionează capul de citire-scriere pe prima celulă din stânga şi se urmează instrucţiunile „programului" (de înmulţire) până se ajunge într-o *stare finală*, iar maşina se *opreşte*, nu mai există nicio instrucţiune care să poată fi urmată. Conţinutul benzii la acel moment este rezultatul calculului.

Extrem de reducţionist, dar acesta este cel mai general model de calcul algoritmic – în lipsa unei definiţii anterioare a ceea ce este calculabil, afirmaţia este doar o ipoteză şi ea poartă numele de *teză Turing-Church*. Dar, ceea ce a făcut maşina Turing atrăgătoare au fost nu numai simplitatea şi puterea sa (s-a demonstrat că poate simula orice alt model de calcul), ci şi *robusteţea* (puterea de calcul nu i se schimbă dacă se fac adăugiri în alcătuire sau funcţionare, cum ar fi adăugarea de benzi, prelungirea infinită a benzii şi în partea stângă, funcţionarea nedeterministă) şi, mai ales, existenţa *maşinilor universale*: există o maşină Turing, *TMU*, care poate simula orice maşină Turing particulară, *TM*, în sensul următor. Dacă pe banda maşinii *TMU* scriem un cod al maşinii *TM, cod(TM),* şi date de intrare x ale maşinii *TM*, atunci maşina *TMU* ne va furniza acelaşi rezultat pe care ni-l furnizează *TM* plecând de la datele x. Ceva mai formal (dar omiţând totuşi unele detalii), avem $TMU(cod(TM),x) = TM(x)$. Iar Turing a demonstrat că există maşini Turing universale. Se întâmpla în 1936, în lucrarea „On computable numbers, with an application to the Entscheidungsproblem", publicată în *Proceedings of the London Mathematical Society,* Ser. 2, vol. 42, 1936, pp. 230-265, cu o erată în vol. 43, 1936, pp. 544-546.

Avem aici „certificatul de naştere" al calculatoarelor de azi, numite de aceea şi de tip Turing-von Neumann (atunci când a realizat primele calculatoare electronice

programabile, la mijlocul anilor 1940, John von Neumann a fost în mod esențial influențat de ideile lui Turing).

Câteva lucruri merită a fi subliniate: codul maşinii *TM* este programul de simulat/executat pe *TMU*, plecând de la datele *x*; instrucțiunile maşinii *TMU* constituie „sistemul de operare" al calculatorului; datele şi programul sunt plasate în acelaşi loc (pe banda *TMU*, în memoria calculatorului) – de aici posibilitatea de a le procesa în acelaşi fel, deci vulnerabilitatea la viruşi.

Important pentru calculul natural: activitatea maşinii Turing este secvențială, la fiecare pas se execută o singură instrucțiune. În multe locuri din natură, dacă nu în majoritatea, în particular, din biologie, procesele sunt paralele, ceea ce constituie o mare atractivitate pentru informatică, dar nu sunt neapărat sincronizate, ceea ce ridică probleme pentru informatică.

Mai sunt şi alte diferențe între maşina Turing, „calculatoarele biologice" şi calculatoarele electronice, pe care le voi aminti la momentul potrivit.

Reținem deocamdată că, în cele ce urmează, a calcula are înțelesul sugerat de maşina Turing: există o intrare şi o ieşire, iar între ele există un algoritm care face trecerea de la intrare la ieşire, rezultatul unui calcul fiind obținut în momentul în care maşina se opreşte. Restrictiv, dar precis. Având un asemenea cadru la îndemână, putem căuta calcule în jur, mai mult, le putem studia într-un context foarte bine dezvoltat, teoria calculabilității – un ansamblu, de fapt, de mai multe teorii, precum teoria automatelor, teoria limbajelor formale (gramaticilor), teoria complexității calculului şi altele.

Informatică şi matematică

Poate că acesta este locul în care să amintim încă o discuție cu variate luări de poziție, legată de relația dintre informatică şi matematică. Ea a apărut şi în Academie, apare în învățământul superior (prin anii '60–'70, la vremea sputnicilor şi hidrocentralelor, multe facultăți erau „de matematică-mecanică", acum mecanica a fost înlocuită cu informatica), chestiunea este dezbătută uneori şi în presă. De fapt, contextul este mai larg, punându-se uneori sub semnul întrebării relația matematicii cu celelalte ştiințe, cu şcoala, cu societatea. Există persoane care se mândresc cu faptul că „nu au fost buni la matematică", există opinia că matematica este un lux, un „fetiş național" (sintagma a apărut de curând în titlul unui articol de ziar dedicat subiectului), pe scurt, că se face prea mult caz de matematică şi, deci, se face prea multă matematică. Iar acest curent de opinie capătă extindere, ajutat şi de informatică („nu trebuie să mai ştim tabla înmulțirii, o ştie calculatorul pentru noi").

Sigur, există o problemă cu educația matematică. *Cât, ce* şi, mai ales, *cum*? – şi mai sunt şi alte întrebări; le putem găsi, adesea şi cu soluții, în scrierile din ultimii ani ale profesorului Solomon Marcus dedicate educației. Problema nu se poate însă soluționa de jos în sus, matematicienii din cercetare şi din învățământul superior sunt cei care trebuie să şi-o asume – o spune, de pildă, Juraj Hromkovic, de la ETH Zürich, într-un articol din revista *Curtea de la Argeş* (www.curteadelaarges.ro, august 2014), bazându-se pe activitatea practică în această direcție a centrului unde lucrează (activitate materializată, printre altele, în manuale de matematică de un tip nou). Matematicienii trebuie să fie şi cei care să iasă în public şi să pledeze pentru matematică, prima vină este a lor dacă domeniul îşi pierde din popularitate. Pentru matematician, matematica este un mare joc, care, ca orice joc, are recompensa intrinsecă, în însăşi buna sa desfăşurare, prin urmare, este explicabil ca interesul pentru „popularizare" să fie scăzut,

dar cei care-şi laudă infirmitatea matematică, fie ea reală sau pretinsă, sunt de obicei mai vizibili, mai vocali, iar pericolul este evident.

Rămânând la relaţia dintre matematică şi informatică, să consemnăm că informatica teoretică, aflată la intersecţia celor două ştiinţe, este adesea considerată parte a matematicii de către informaticieni şi parte a informaticii de către matematicieni, având uneori probleme şi în cadrul informaticii – de altfel, ca orice ramură teoretică a unei ştiinţe cu mare pondere practică. Evident, false probleme în sine, dar care pot avea urmări administrative neplăcute.

Apelez, fiind de aceeaşi părere, la o voce autorizată, Edsger W. Dijkstra, un clasic al informaticii, ba chiar al celei practice: ajunge să menţionez că prin anii ’60 a lucrat la implementarea limbajului Algol la Centrul de Matematică din Amsterdam şi tot el este promotorul *programării structurate*, care a făcut epocă printre creatorii de software. (Poate ar trebui să adaug aici că primii patru ani după absolvirea facultăţii am programat din plin, în Cobol şi Fortran, ba chiar am realizat programele pentru plata salariilor la o mare uzină bucureşteană – îmi amintesc, aşadar, ce înseamnă informatica practică...)

„Sfârşitul informaticii?”, se întreabă Dijkstra retoric-ironic în titlul unei note apărute în *Communications of the ACM* (vol. 44, martie 2001, p. 92), care începe cu următoarea frază: „În universităţi, industrie şi lumea comercială, există o părere răspândită cum că informatica în sine s-a încheiat şi, deci, ea s-a maturizat, trecând de la un subiect teoretic, pentru cercetători, la o chestiune practică, pentru ingineri, manageri şi antreprenori.” Apoi: „Această părere este corectă numai dacă identificăm obiectivele informaticii cu ceea ce s-a obţinut deja şi uităm acele obiective pe care nu am reuşit să le atingem, chiar dacă ele sunt prea importante pentru a fi ignorate.”

Mult mai explicit este Dijkstra în discursul pe care l-a susţinut în mai 2000, la un simpozion (*In Pursuit of Simplicity*) organizat la Universitatea din Austin-Texas cu prilejul retragerii sale. Titlul discursului (publicat în *Information Processing Letters*, vol. 77, febr. 2011, pp. 53-61) este semnificativ: „Under the spell of Leibniz’s dream”. Reiau câteva fraze-aforism:

„Ceea ce este frumos teoretic tinde să fie profund util. (...) În proiectarea de sisteme digitale sofisticate, eleganţa nu este un lux de care ne putem dispensa, ci o chestiune de viaţă şi de moarte, un factor major care decide între succes şi eşec.”

„În zilele noastre, există o asemenea obsesie a aplicaţiilor că, dacă universităţile nu sunt atente, forţele externe, cele care fac distincţia [dintre teorie şi practică], vor trage o linie între cele două şi pot încerca să-i surghiunească pe teoreticieni în ghetoul unor departamente separate şi în clădiri separate. O simplă extrapolare ne spune că, după o vreme, practicienii vor avea puţine de aplicat; lucrul acesta este bine cunoscut, dar nu a împiedicat niciodată minţile contabile să omoare gâsca cu ouă de aur. Cel mai rău lucru cu institutele explicit dedicate aplicării ştiinţei este că ele tind să devină institute dedicate teoriei de mâna a doua.”

Pledoaria pentru a ne aşeza sub îndemnul/vraja lui Leibniz este evidentă, pentru că „simbolurile direcţionează raţiunea”, iar, după ce vom avea un limbaj în care „toate adevărurile raţiunii se vor reduce la un fel de calcul”, „erorile vor fi doar greşeli de calcul”. (E adevărat, programul lui Leibniz, preluat şi formulat în termeni mai precişi de David Hilbert, nu poate fi realizat; pe de o parte, matematica este prea exact-riguroasă iar realitatea prea complexă şi nuanţată pentru a putea transforma totul în calcule, pe de altă parte, teoremele lui Gödel au arătat că nici programul lui Hilbert nu este realizabil.)

Bineînţeles, relaţia matematicii cu informatica este mult mai complexă, dar nu este aici locul pentru a o explora în mai multe amănunte. Închei întorcându-mă în

punctul din care am plecat: calculatoarele de azi, programabile, de tip Turing-von Neumann, s-au născut din teorema de universalitate a lui Turing, din 1936. Interesant (şi reconfortant pentru poziția lui Dijkstra) este că, în urma unui vot pe internet, din 2013, menit a identifica cea mai importantă descoperire ştiinţifică sau tehnologică britanică, locul întâi a fost ocupat, surprinzător pentru vremurile noastre pragmatice, de maşina Turing şi teorema de universalitate a lui Turing, care au câştigat competiția cu linoleumul, motorul cu abur, telefonul, cimentul, fibra de carbon şi altele asemeni.

Calculează natura?

Având în minte calculabilitatea în sens Turing, întrebarea devine şi mai restrictivă, dar discuția dinainte ne oferă limitele între care să căutăm răspunsul: da, cel puțin la nivelul... oamenilor, respectiv, da, oriunde există un proces care poate fi interpretat drept calcul de către un observator potrivit. Opinii care se plasează mai aproape de prima sau de a doua dintre aceste limite pot fi găsite uşor, citez doar una din extremitatea foarte permisivă, ba chiar trecând peste ea, pentru că nici observatorul nu mai este invocat.

La începutul capitolului 2 („Molecular Computation") al volumului colectiv *Non-Standard Computation* (T. Gramss, S. Bornholdt, M. Gross, M. Mitchel, Th. Pellizzari, eds., Wiley-VCH, Weinheim, 1998), M. Gross spune: „Viața este calcul. Fiecare celulă în parte citeşte informație dintr-o memorie, o rescrie, primeşte date de intrare (informație asupra mediului), procesează date şi acționează conform rezultatelor tuturor acestor calcule. Pe total, zilioanele de celule care populează biosfera efectuează cu siguranță mai mulți paşi de calcul pe unitatea de timp decât toate calculatoarele oamenilor puse la un loc."

În cele ce urmează, adopt o poziție mai conservatoare şi mai productivă: având în minte definiția matematică a calculabilității, în speță, abordarea de tip Turing, să privim în jur, în special în biologie, în căutare de idei, suporturi de date, structuri de date, operații cu acestea, moduri de control al operațiilor, arhitecturi de „calculatoare", care pot sugera (1) noi modele de calcul, (2) căi de folosire mai bună a calculatoarelor existente, (3) posibilități de îmbunătățire hardware a calculatoarelor existente, eventual chiar (4) noi tipuri de calculatoare, bazate pe materiale de proveniență biologică. A se observa ambiția în creştere de la un obiectiv la altul. De remarcat că în calculul cu ADN s-a încercat încă de la început realizarea de calcule în eprubetă, adresându-se astfel, direct, obiectivul al patrulea.

Am avut aici în vedere în primul rând interesul informaticii, dar primul obiectiv acoperă şi cea de a doua direcție de cercetare menționată în prefața handbook-ului de calcul natural, investigarea în termeni informatici a proceselor care au loc în natură, iar această direcție ar trebui subliniată explicit, separat, mai ales pentru a anunța un „efect colateral" al demersului, anume reîntoarcerea la biologie, furnizarea de modele utile biologului.

La ora aceasta, calculul cu ADN a fost foarte puțin de ajutor informaticii (practice), puțin de ajutor biologiei şi mult folositor nanotehnologiei, prin sugerarea unor noi probleme şi teme de cercetare, în timp ce calculul membranar are aplicații semnificative şi în informatică şi în biologie, cu promisiuni poate chiar mai mari în cel de-al doilea domeniu, inclusiv în biomedicină şi ecologie.

O precizare: „interesul informaticii" acoperă şi interesul teoretic, care nu trebuie să ducă neapărat la aplicații, în sensul imediat al termenului. Să ne gândim, de exemplu, la ciliate. În procesul de diviziune, când trec de la genele micronucleare la cele macronucleare, aceste unicelulare duc la bun sfârşit operații complicate de tip procesare

de liste, şi asta de milioane de ani, cu mult înainte ca informaticienii să dea nume şi să studieze această structură de date. Evident, ciliatele nu la calcule se gândesc în timp ce fac acest lucru, dar, noi, oamenii, putem construi teorii de toată frumusețea plecând de aici, inclusiv modele de calcul, echivalente uneori cu maşina Turing. Trimit pentru referințe la monografia *Computation in Living Cells. Gene Assembly in Ciliates* (Springer-Verlag, 2004), de A. Ehrenfeucht, T. Harju, I. Petre, D.M. Prescott şi G. Rozenberg.

O dilemă eternă

Discuția anterioară ne împinge inevitabil spre lunga dezbatere privind relația dintre invenție şi descoperire. Bibliografia este imensă, citez doar cartea-bibliotecă a acad. Solomon Marcus *Invenție şi descoperire*, Cartea Românească, 1989. Cât este invenție şi cât descoperire în informatică – cu particularizare în calculul natural? Nu încerc un răspuns, vor fi atâtea câte puncte de vedere, câte experiențe personale şi câte situări filosofice. Modelele cu care lucrăm sunt de natură matematică, punctul de vedere platonician ne asigură că totul este descoperire, pentru că matematica însăşi este realitate revelată. Da, dar s-a convenit că noțiunile, conceptele, teoriile şi modelele sunt inventate, teoremele descoperite, demonstrațiile inventate. Pentru a continua alternanța, putem adăuga că aplicațiile sunt descoperite. Modelele sunt, deci, considerate invenții.

Aş introduce însă o nuanță. Modelele se aşază pe structuri care există, dar nu au primit încă nume. În plus, spre deosebire de un zid poate fi descoperit şi de un arheolog care ştie ce caută şi de un pomicultor care sapă cu alte motive decât descoperirea unei fundații de biserică, un model de calcul nu poate fi „văzut" într-o celulă decât de un informatician care are deja în minte modele de calcul. De pildă, procesele numite de biologi simport şi antiport există, funcționează dintotdeauna în modul lor ingenios, dar ele nu *calculează* decât pentru matematicianul care caută un model de calcul bazat pe trecerea de „obiecte" dintr-un compartiment al celulei în altul. „Calcul prin comunicare", am căutat o vreme aşa ceva, intuind că există, şi am avut rezolvarea atunci când un biolog (Ioan Ardelean) mi-a vorbit despre operațiile de simport şi antiport. Un model mai degrabă descoperit decât inventat. Dar, o descoperire care nu s-a făcut prin scoaterea la lumină a obiectului descoperit, ci prin suprapunerea pe o felie de realitate a unui model intuit, similar unor modele existente deja şi actualizat în dialogul dintre limbajul formal şi realitate. Invenție şi descoperire în acelaşi timp.

O altă nesfârşită discuție

Nu continui cu alte întrebări la fel de fluide, menite a-şi păstra interesul în ciuda oricâtor răspunsuri. (De pildă, dându-ne prilejul de a ne întreba cât este ştiință şi cât artă în informatică, Donald Knuth îşi intitula un impresionant tratat, planificat a apărea în multe volume, *Arta programării calculatoarelor*.) Ating însă un alt subiect sensibil, cu care am fost confruntat uneori sub forma întrebării gazetăreşti, dar nu lipsite de noimă: „În cercetările dumneavoastră legate de celulă, v-ați întâlnit cu Dumnezeu?" Evident, se aşteaptă altceva decât un răspuns de tipul „da" sau „nu" şi la fel de evident este că, dacă luăm întrebarea în serios, eşuăm pe nisipurile mişcătoare ale opțiunilor personale, credinței, metaforei.

Dacă Dumnezeu este ordinea, organizarea, binele, „Dumnezeul lui Spinoza, care se manifestă prin armonia legilor universului", cum ar spune Einstein, atunci da, Îl întâlnesc continuu, şi în celule şi în afara lor. Mai mult: în titlul unei cărți traduse la Humanitas, în 2011, Mario Livio se întreabă *Este Dumnezeu matematician?* Răspund în

stil platonician: nu, Dumnezeu nu este matematician, El este chiar matematica („gramatica" lumii) – deci, iarăşi, mă întâlnesc zilnic cu El.

Dacă însă Dumnezeu este ceea ce-mi propune Cartea, atunci mă aliniez lui Galilei, care, într-o scrisoare către don Benedetto Castelli, de la 21 decembrie 1613, spunea (reiau după Edmond Constantinescu, *Dumnezeu nu joacă zaruri*, Editura MajestiPress, Arad, 2008; din păcate, scrisoarea respectivă nu apare în volumul Galileo Galilei, *Scrisori copernicane*, Humanitas, 2010): „Dumnezeu a scris două cărți, Biblia şi Cartea naturii. Biblia a fost scrisă în limbaj omenesc. Cartea naturii a fost scrisă în limbajul matematicii. De aceea, limbajul Bibliei este nepotrivit pentru a vorbi despre natură. Cele două trebuie studiate independent una de alta." Cartea naturii, mai adăuga Galilei, ne învață „cum merge Cerul", Biblia ne învață „cum să mergem la Cer".

După secole de separări – mai degrabă dogmatice, de ambele părți –, alternând cu încercări – multe dintre ele patetice – de reconciliere a ştiinței cu religia, spusele lui Galilei pot părea simpliste sau conformiste, dar ele taie cu eficiență un mereu regenerat nod gordian. Consemnez, la extrema sofisticată, situația oarecum simetrică a lui Francis S. Collins, nu numai contemporan cu noi, dar şi legat de subiectul discursului de față: el a fost director al National Human Genome Research Institute, liderul faimosului proiect al genomului uman. În 2006, a publicat o carte mult discutată, tradusă în 2009 la Editura Curtea Veche, intitulată *Limbajul lui Dumnezeu*. Sintagma a fost folosită şi de Bill Clinton, atunci când, în 2001, a anunțat completarea „celei mai importante şi mai impresionante hărți întocmite vreodată de omenire", secvențierea celor în jur de trei miliarde de „litere" ale „cărții vieții". Chiar dacă titlul ar sugera asta, Collins nu este însă nici creaționist şi nici adept al proiectării inteligente, ci „teist evoluționist", iar concluzia sa este că „Dumnezeul Bibliei este şi Dumnezeul genomului" (p. 222 în cartea citată), iar „ştiința poate fi o formă de religiozitate" (pag. 240). O altă poziționare confortabilă – dar în cele ce urmează rămân alături de Galilei...

Limitele calculatoarelor actuale

Moda calculului natural, cu deosebire a celui inspirat din biologie, nu are numai motivația interioară, a multelor direcții de cercetare explorate în ultimele decenii şi dovedite a fi interesante teoretic şi utile, sau măcar promițătoare, practic, ci are şi o motivare exterioară, legată de limitele calculatoarelor actuale, unele dintre ele resimțite tot mai acut. Calculatoarele sunt invenția cu cel mai mare impact a secolului XX, cu implicații resimțite în toate componentele vieții noastre, din comunicare în funcționarea sistemului bancar, din sănătate în armată, de la numeroasele gageturi care ne înconjoară la internet. Cu toate acestea – de fapt, tocmai pentru asta – calculatoarele pe care le avem acum au limitări de care ne lovim tot mai des (cu mențiunea că există şi aici, ca în mai toate lucrurile, ceva bine în rău şi ceva rău în bine: calculatoare puternice pot fi folosite şi în scopuri bune, dar şi în scopuri rele – cum ar fi spargerea de servere, de sisteme de criptare, pe care se bazează comunicarea protejată). Să gândim însă pozitiv şi să notăm că există multe lucruri pe care calculatoarele nu le pot face încă, dar pe care ne-ar plăcea, sau chiar folosi, să le facă.

Procesoarele devin tot mai rapide şi mai compacte, capacitatea memoriei lor tot mai largă. Da, dar până când? A fost mult repetată aşa-numita *lege a lui Moore*, enunțată în 1965 de Gordon A. Moore, cofondator al Intel Corporation, cu referire la numărul tranzistorilor de pe un circuit integrat, extinsă apoi la costul pe unitatea de informație memorată, formulată uneori şi în forma „în fiecare an, calculatoarele devin de două ori mai mici, de două ori mai ieftine şi de două ori mai puternice". Exponențial pe fiecare dintre cele trei dimensiuni, tinzând deci rapid spre limita cuantică în

dimensiunea procesoarelor. Chiar la nivelul mai tehnic, confirmat pentru câteva decenii, al dublării capacității procesoarelor, legea – de fapt, o observație, urmată de o previziune – a fost ajustată, cu dublarea prevăzută mai întâi la un an şi jumătate, apoi la trei ani. Încă nu e rău, dar nu va putea continua prea mult nici în acest ritm.

Problema e însă alta. Progrese se fac continuu şi se vor mai face la nivel tehnologic, dar calculatorul actual are limite principiale, peste care nu poate trece numai prin progrese tehnologice. Recunoaşte amprente, dar nu fețe de persoane, joacă şah la nivel de campion mondial, dar (pe tabla standard, nu pe table reduse) GO joacă numai la nivel de începător, demonstrează teoreme de calculul propozițiilor, dar nu trece mult peste acest nivel (şi nici nu distinge între o teoremă trivială şi neinteresantă şi una care merită a fi pusă pe hârtie). Toate acestea şi multe altele, în primul rând pentru că este calculator... Turing-von Neumann. Secvențial adică. Uniprocesor. (Mai are şi alte slăbiciuni, mai puțin restrictive în aplicațiile curente – de pildă, este un destul de mare consumator de energie.) Calculează tot ce se poate calcula, dar asta *în principiu*, la nivelul competenței. Întâlnim şi un aspect istoric aici. La început ne-a interesat ce se poate calcula, unde este frontiera calculabilității, a decidabilității algoritmice. Chestiuni fundamentale matematic, dar în aplicații devine relevantă *performanța*, resursele cerute de un calcul dat, ce putem calcula acum şi aici, în condiții precizate. Câtă electricitate consumă un calculator sau cât spațiu ocupă nu mai sunt întrebări de interes curent, aşa cum erau prin anii ’60 (sunt încă de interes în condiții speciale, în cosmos, robotică), dar în cât timp primim răspunsul la o problemă sau rezultatul unui calcul este un aspect crucial în orice aplicație. Şi, spuneam, nu promisiunea tehnologică este relevantă aici, ci abordarea matematică, frontiera teoretică dintre fezabil şi nefezabil.

O mare provocare: complexitatea exponențială

O puternică teorie s-a dezvoltat asupra acestui subiect, teoria complexității calculului, care a definit încă de la început ca fiind fezabile problemele care se pot rezolva în timp polinomial în raport cu dimensiunea datelor de intrare. (Exemplu: avem un graf – o hartă cu localități şi şosele între ele – cu n noduri. În cât timp poate un algoritm să ne spună dacă graful conține un drum hamiltonian, adică unul care trece prin toate nodurile, o singură dată prin fiecare? Dacă acest timp este mărginit de un polinom în n, atunci spunem că algoritmul este de complexitate polinomială.) Problemele de complexitate exponențială, care cer un timp de tip 2^n, 3^n etc. pentru o intrare de dimenisune n, au fost considerate nefezabile. Prima clasă a fost notată cu **P**, a doua cu **NP**, abrevierile venind de la „polinomial”, respectiv, „polinomial nedeterminist”: o problemă este în **NP** dacă putem decide în timp polinomial dacă o soluție propusă pentru ea este cu adevărat soluție sau nu („ghicim” o soluție, apoi verificăm dacă ea este corectă; mai tehnic, soluția este găsită de o maşină Turing nedeterministă, care are mai multe alternative la un pas de calcul şi contăm pe faptul că alege totdeauna calea cea bună, fără a examina exhaustiv posibilitățile). Pentru detalii, se poate consulta, de exemplu, cartea clasică a lui C.H. Papadimitriu, *Computational Complexity,* Addison-Wesley, 1994.

Să amintim şi că în clasa **NP** există o subclasă, a „celor mai grele probleme din **NP**”, problemele **NP**-*complete;* o problemă este de acest tip dacă orice altă problemă din **NP** se poate reduce la ea în timp polinomial. Prin urmare, dacă o problemă **NP**-completă s-ar putea rezolva în timp polinomial, atunci *toate* problemele din **NP** ar avea o astfel de rezolvare. Problemele folosite în criptografie sunt de obicei **NP**-complete.

O teorie foarte frumoasă, care, în varianta de bază, are însă trei slăbiciuni: (1) încă nu ne poate spune dacă **P** = **NP**, dacă nu cumva soluții polinomiale se pot găsi şi

pentru problemele pe care acum le considerăm de complexitate exponențială, (2) teoria nu ține seama de „detalii” cum ar fi coeficienții şi gradul polinoamelor şi care, practic, pot avea o influență crucială asupra timpului de calcul, şi (3) teoria ia în seamă cazurile extreme, este de tip *worst case*, numără paşii de rezolvare a celor mai dificile probleme dintr-o clasă, în timp ce realitatea se plasează de cele mai multe ori la mijloc, spre „medie”. Iată un exemplu cu relevanță practică: problemele de programare liniară sunt în **P**, pentru că algoritmul elipsoidal al lui Leonid Khachian (1979) rezolvă problema în timp polinomial, dar polinomul cu pricina este atât de complex, încât în practică, în majoritatea cazurilor, acest algoritm este mai puțin eficient decât vechiul algoritm simplex, propus în vremea celui de-Al Doilea Război Mondial şi considerat unul dintre primii zece cei mai importanți algoritmi imaginați vreodată, chiar dacă el este, teoretic, de complexitate exponențială.

Din aceste motive, teoria complexității s-a rafinat şi diversificat (complexitate medie, algoritmi aproximativi – ultimii având o directă legătură cu calculul natural), iar definiția fezabilității s-a redefinit cu mai multă grijă.

Oricum, convingerea este transformată în slogan: *calculatoarele Turing-von Neumann nu pot rezolva în timp rezonabil probleme de complexitate exponențială.*

Interesul față de problema dacă **P** = **NP** este enorm. Pe de o parte, majoritatea problemelor practice (netriviale) sunt în clasa **NP**, deci sunt nefezabile, nu pot fi rezolvate eficient, pe de alta, majoritatea sistemelor criptografice în uz se bazează pe probleme de complexitate exponențială, rezolvarea lor polinomială putând duce la spargerea sistemelor. Problema este deschisă încă din 1971 (a fost formulată de Stephen Cook), iar în anul 2000 a fost inclusă de Clay Mathematical Institute, Cambridge, Massachusetts, în lista celor şapte „probleme ale mileniului”, rezolvarea ei fiind recompensată cu un milion de dolari.

Dacă importanța problemei pentru informatica teoretică nu poate fi supraestimată, nu este clar care ar putea fi urmarea practică a unei soluții, indiferent care ar fi aceasta. Există multe discuții pe această temă – a se vedea, de exemplu, S. Cook, „The importance of the **P** versus **NP** question”, *Journal of the ACM*, vol. 50, 2003, pp. 27-29. Dacă se va arăta că **P** este strict inclus în **NP**, aşa cum majoritatea informaticienilor (dar nu toți!) cred, atunci nimic nu se va schimba la nivelul informaticii practice. Dacă egalitatea se va demonstra neconstructiv sau constructiv, dar într-o manieră nefezabilă (vor fi găsite soluții polinomiale la probleme din **NP**, dar cu polinoame de grade mari sau cu coeficienți uriaşi), urmările practice vor fi nesemnificative (dar va începe cursa pentru găsirea unor soluții ad-hoc, pentru care timpul să fie măsurat de polinoame mai cumsecade). Dacă, însă, va fi obținută o trecere necostisitoare de la **NP** la **P**, atunci urmările pentru informatica practică pot fi spectaculoase – în bine, exceptând criptografia, unde urmările vor fi dramatice.

La nivelul software-ului în uz, mai există o problemă, pe care o menționez în formularea lui Edsger W. Dijkstra (din „The end of computer science?”, articolul citat mai devreme): „Cele mai multe dintre sistemele noastre [de programe] sunt mult mai complicate decât nivelul la care ar putea fi considerate sănătoase, sunt prea haotice pentru a fi folosite cu încredere şi confortabil. Clientul mediu al industriei informatice a fost servit atât de prost, că el se aşteaptă în fiecare moment ca sistemul lui să cadă.” Lipsa de robustețe a sistemelor informatice complexe este o preocupare la fel de actuală şi azi precum era în anul 2000.

Pentru a ilustra faptul că nu prin progrese tehnologice putem face față complexității exponențiale, să examinăm un caz simplu: să considerăm o problemă a cărei complexitate este de tip 2^n, o problemă de grafuri, de exemplu, care poate fi

rezolvată de un calculator obişnuit pentru, să zicem, 500 de noduri, în aproximativ un sfert de oră; să presupunem că tehnologia ne oferă un calculator de 1000 de ori mai rapid, un pas înainte cu totul netrivial şi nu tocmai frecvent. Este adevărat că problemele pe care le rezolvam mai devreme într-un sfert de oră (aproximativ o mie de secunde) se vor rezolva acum într-o secundă, dar, dacă încercăm grafuri cu mai mult de 500 de noduri, constatăm că progresul este derizoriu: cu noul calculator, vom rezolva problema, într-un sfert de oră, numai pentru grafuri cu cel mult 510 noduri. Asta, pentru simplul motiv că 2^{10} este deja mai mare de 1000. Iar dacă problema ar fi de complexitate 3^n, atunci ne-am opri pe la 506 noduri...

Promisiunile calculului natural

Pentru a face faţă problemelor de complexitate exponenţială, dar şi pentru alte motive pe care le voi aminti mai târziu, informatica are mai multe posibile soluţii, majoritatea legate de calculul natural, ba chiar de biocalculabilitate: (1) căutarea de calculatoare masiv paralele, (2) căutarea de calculatoare/calcule nedeterministe, (3) căutarea de soluţii aproximative, (4) căutarea de soluţii probabiliste.

Toate aceste patru direcţii au fost explorate şi în cadrul informaticii „standard", atât în teorie, cât şi în tehnologia de tip electronic. Calculatoare multiprocesor sunt deja disponibile de mai mulţi ani – dar fără a ajunge la paralelismul masiv de la care se aşteaptă rezolvarea problemelor complexe. Dacă un număr mare de procesoare sunt puse împreună, apar alte probleme, unele inginereşti (disiparea de căldură), altele, mai importante poate, teoretice, legate de sincronizarea procesoarelor. O ramură distinctă a complexităţii calculului se ocupă de complexitatea sincronizării – a se vedea, de exemplu, monografia lui Juraj Hromkovic, *Communication Complexity and Parallel Computing,* Springer-Verlag, 1997. Una dintre concluziile acestei teorii spune că, de la un număr de procesoare în sus, sincronizarea (măsurată prin numărul de biţi necesari în acest scop) devine mai costisitoare decât calculul propriu-zis, ceea ce sugerează renunţarea la sincronizare, dar de aici apar alte probleme, pentru că nu suntem încă obişnuiţi cu programarea de calculatoare asincrone.

Cu atât mai puţin ştim cum să construim şi să folosim „calculatoare nedeterministe". Foarte atrăgătoare sunt însă ultimele două dintre cele patru soluţii şi aici intervine „forţa brută" a calculatoarelor. Ea se aplică mai ales problemelor complexe de optimizare: explorând la întâmplare spaţiul soluţiilor posibile, suficient de mult timp, cu o mare probabilitate putem da peste soluţii optime sau aproape de cele optime. Soluţii aproximative, eventual cu o probabilitate cunoscută de a fi optime.

Aici intră promiţător în scenă calculul natural. Mă voi referi de acum înainte numai la cel de inspiraţie biologică.

Într-o celulă, un număr imens de „chimicale" (ioni, molecule simple, macromolecule, proteine, molecule de ADN şi ARN) evoluează împreună, în soluţie apoasă, la un înalt nivel de paralelism şi, în acelaşi timp, de nedeterminism, într-un mod robust, coordonat, făcând faţă cu succes solicitărilor care vin dinspre mediu şi căpătând în timp caracteristici foarte atrăgătoare, de tip adaptare, învăţare, autoreparare, reproducere. Multe alte detalii sunt de interes, cum ar fi reversibilitatea unor procese sau eficienţa energetică, cu numărul de operaţii per Joule mult mai ridicat decât în cazul procesării electronice a informaţiei (ştergerea consumă energie, de aceea sunt de interes calculatoare reversibile, vezi R. Landauer, „Irreversibility and heat generation in the computing process", *IBM Journal of Research and Development,* vol. 5, 1961, pp. 183-191, şi C.H. Bennett, „Logical reversibility of computation", *Idem*, vol. 17, 1973, pp. 525-532).

Se pare, deci, că natura a perfecționat de-a lungul a patru miliarde de ani de evoluție procese (şi suporturi materiale pentru acestea) care aşteaptă să fie identificate şi înțelese de informaticieni, pentru a învăța noi metode şi paradigme de calcul, eventual pentru a construi noi tipuri de calculatoare. Iar informaticienii au trecut de multă vreme la lucru.

Doar câțiva paşi pe acest drum, pe scurt: *Algoritmii genetici*, ca mod de a organiza căutarea prin spațiul soluțiilor, imitând evoluția darwiniană, pentru a rezolva probleme de optimizare. Generalizarea la *calculul evolutiv* şi *programarea evolutivă. Rețelele neurale*, încercând să imite funcționarea creierului uman, tot pentru căutarea de soluții aproximative, mai ales pentru probleme de recunoaştere a formelor, de învățare. Ceva mai târziu, *calculul cu ADN*, care a propus şi un nou hardware, masiv paralel, folosind ca suport de calcul molecula de ADN. Şi mai aproape de noi, *calculul membranar*, luând ca punct de plecare celula ca atare şi populațiile de celule.

La rândul său, calculul evolutiv, în general, zona algorimilor aproximativi inspirați din biologie, s-a ramificat spectaculos, în direcții dintre cele mai diverse (şi, uneori, pitoreşti; păstrez denumirile englezeşti): *immune computing, ant colony algorithm, bee colony algorithm, swarm computing, water flowing computing, cultural algorithm, cuckoo algorithm, strawberry algorithm* – şi e foarte probabil să mai fi apărut şi altele între timp...

Este important să subliniem faptul că toate ramurile calculului natural menționate mai devreme, cu excepția calculului cu ADN, sunt menite implementării pe calculatorul obişnuit, spre mai eficienta lui folosire, ele propun noi tipuri de software, nu modificarea arhitecturii, a hardware-ului.

Totul începe cu Turing

Într-un anume sens şi într-o anume măsură, întreaga istorie a informaticii (teoretice) este legată de biologie, a căutat şi a găsit surse de inspirație în aceasta. Am menționat deja că Turing, în 1935-1936, când a introdus maşina care-i poartă numele, a încercat să simuleze modul uman de a calcula.

După un deceniu, McCullock, Pitts, Kleene au fundamentat teoria automatelor finite plecând de la modelarea neuronului şi a rețelelor de neuroni. Ceva mai târziu, acelaşi punct de plecare a condus la ceea ce astăzi este numit calcul neural.

Interesant este că începuturile acestei discipline pot fi identificate în lucrări nepublicate ale aceluiaşi Alan Turing. Avem aici un episod interesant, care poate ilustra influența psihologiei şi sociologiei asupra ştiinței, vorbind despre lideri de grup neinspirați şi despre cercetători interesați mai mult în cercetarea lor decât în publicarea rezultatelor. Anume, în 1948, Turing a scris o scurtă lucrare, intitulată „Intelligent machinery", care a rămas nepublicată până în 1968, deoarece şeful său de la National Physical Laboratory din Londra, ironic, pe nume sir Charles Darwin, nepotul celebrului naturalist omonim, a scris pe colțul lucrării „schoolboy essay", eseu şcolăresc, oprind publicarea.

„Această lucrare a fost primul manifest al inteligenței artificiale. În lucrare (...), matematicienul britanic nu numai că a pus bazele conecționismului, dar a şi introdus, într-un mod strălucitor, multe dintre conceptele care ulterior au devenit centrale în inteligența artificială, în unele cazuri după ce au fost reinventate de alții..." – am citat din B.J. Copeland, D. Proudfoot, „Alan Turing's forgotten ideas in computer science", *Scientific American*, aprilie 1999, pp. 77-81. Printre altele, lucrarea lui Turing introduce două tipuri de rețele de „neuroni" conectați aleatoriu, ca un pas înspre crearea unei maşini inteligente, una dintre trăsăturile cheie ale acestor rețele fiind aceea de a învăța,

de a se antrena în vederea rezolvării de probleme. Este calcul neural *avant la lettre*, cu ideile redescoperite ulterior, fără referire la Turing. Detalii despre „maşinile neorganizate" ale lui Turing pot fi găsite şi în C. Teuscher, ed., *Alan Turing. Life and Legacy of a Great Thinker,* Springer-Verlag, 2003, şi C. Teuscher, E. Sánchez, „A revival of Turing's forgotten connectionist ideas: exploring unorganized machines", din *Proc. Connectionist Models of Learning, Development and Evolution*, Liège, Belgium, 2000 (R.M. French, J.J. Sougne, eds.), Springer-Verlag, 2001, pp. 153-162. De asemenea, la adresa http://www.AlanTuring.net se pot afla mai multe despre manuscrisele nepublicate ale lui Turing şi despre eforturile recente de valorificare a lor.

Acelaşi Turing, în acelaşi an, 1948, a propus şi „căutarea genetică sau evolutivă", primele idei ale calculului evolutiv de mai târziu, domeniu care sintetizează acum mai multe ramuri (re)lansate de-a lungul anilor: programare evolutivă (L.J. Fogel, A.J. Owens, M.J. Walsh), algoritmi genetici (J.H. Holland), strategii evolutive (I. Rechenberg, H.P. Schwefel), toate trei iniţiate în anii 1960, şi programarea genetică (J.R. Koza, anii 1990). Primul experiment de „optimizare prin evoluţie şi recombinare" pe calculator a fost efectuat în 1962, de către Bremermann. Detalii pot fi găsite în A.E. Eiben, J.E. Smith, *Introduction to Evolutionary Computing,* Springer-Verlag, 2003.

N-ar fi cu totul surprinzător dacă printre manuscrisele lui Turing s-ar descoperi şi idei legate de calculul cu ADN – să ne amintim că Turing a murit în iunie 1954, iar articolul din *Nature*, în care J.D. Watson şi F.H.C. Crick descriau structura dublu catenară, elicoidală, a moleculei de ADN fusese publicat cu un an înainte („A structure for deoxyribose nucleic acid", vol. 171, aprilie 25, pp. 737-738).

Merită menţionate alte două concepte care fac carieră în informatică şi care ne dau dreptul să spunem că „totul începe cu Turing". Turing însuşi şi-a pus problema şi a propus căi prin care... se poate calcula mai mult decât Turing, imaginând maşini Turing cu *oracol*, subiect iarăşi popular în computabilitatea de azi. De asemenea, el poate fi considerat precursor nu numai al inteligenţei artificiale, ci şi al domeniului numit *viaţă artificială*: în ultimii săi ani, Turing a fost interesat de morfogeneză, de modelarea evoluţiei de la genele unui ovul fertilizat la structura unui animal.

Un exemplu încurajator: algoritmii genetici

Înainte de a trece la calculul cu ADN şi la calculul membranar, subiecte la care voi intra în mai multe detalii, să mai zăbovim puţin asupra unei ramuri a calculului inspirat din biologie care, la prima vedere, este surprinzător de eficient. Este vorba despre *algoritmii genetici*, folosiţi pentru rezolvarea de probleme de optimizare complexe, pentru care nu există algoritmi propriu-zişi (determinişti) sau aceşti algoritmi nu sunt eficienţi. Sloganul subînţeles poate părea derutant: *atunci când nu ştii încotro să mergi, mergi la întâmplare*, cu menţiunea că în cazul algoritmilor genetici „întâmplarea" este direcţionată, ea se face „ca în natură, în evoluţia speciilor".

Totul este o preluare metaforică a unor elemente din evoluţia darwiniană: să presupunem că avem o funcţie de două variabile (reprezentarea ca o suprafaţă de teren, cu văi şi dealuri, este sugestivă) căreia trebuie să-i găsim maximul (unul dintre ele, dacă există mai multe). Dacă nu putem aborda problema analitic, alegem să umblăm la întâmplare prin domeniul de definiţie, căutând maximul. Reprezentăm punctele din domeniu sub forma unor „cromozomi", şiruri binare de lungime constantă. Alegem (la întâmplare sau prin altă metodă) un număr de puncte de plecare, calculăm valoarea funcţiei pentru ele. Trecem apoi la „evoluţie": luăm „cromozomii" doi câte doi şi îi „încrucişăm" (*crossover*), adică îi secţionăm la o poziţie dată şi recombinăm fragmentele, prefixul unuia cu sufixul celuilalt şi invers. Obţinem doi „cromozomi" noi,

descriind doi „indivizi” din „generația” următoare. Calculăm funcția pentru noii „cromozomi”, amestecăm generațiile şi reținem numai o parte dintre indivizi, pe cei cu valorile mai bune ale funcției. Repetăm de un număr precizat de ori, alegem soluția cea mai bună pe care am găsit-o până atunci şi ne oprim.

Nimic nu garantează că ajungem la soluția problemei, că nu ne blocăm, de pildă, într-un maxim local, fără a mai putea ieşi din el, dar, şi aici este surpriza (plăcută), într-un număr semnificativ de aplicații practice, strategia funcționează. Sigur, există nenumărate variante ale scenariului anterior, se şi spune că monografiile din domeniu sunt un fel de „cărți de bucate”, culegeri de rețete, liste de ingrediente şi sugestii de îmbunătățire a algoritmilor: în afară de recombinare, tot ca în evoluție, se foloseşte şi operația de mutație locală, trecerea de la o generație la alta se poate face în multe feluri, populația de „cromozomi” se poate distribui, lucrând local, cu o comunicare de un tip sau altul între regiuni, criteriile de oprire se pot şi ele modifica şi multe altele.

Sunt puse la lucru forța brută a calculatorului, plus metafora evolutivă – cu rezultate, subliniez, neaşteptat de bune: soluții neintuitive, greu de imaginat altfel, convergență rapidă la început, adesea evitând maximele locale. Iar singura „explicație” este cea „bio-mistică”: algoritmii genetici funcționează atât de bine în atât de multe situații pentru că milenii de-a rândul natura a perfecționat procese atât de performante în evoluția speciilor.

Toate acestea induc, la acelaşi nivel speculativ, o concluzie optimistă: dacă algoritmii genetici sunt atât de utili, în ciuda lipsei unui argument matematic pentru aceasta, atunci să încercăm să imităm biologia şi în alte aspecte ale ei, având bune şanse ca, dacă suntem inspirați să extragem ideile potrivite, să obținem sugestii fertile pentru îmbunătățirea folosirii calculatoarelor existente şi, eventual, pentru realizarea unora şi mai performante.

Acest optimism trebuie însă temperat de observația că un rezultat celebru al calculului evolutiv, al domeniului algoritmilor aproximativi în general, este aşa-numita *no free lunch theorem* (teorema „nu există prânz gratuit”), a lui David Wolpert şi William Macready (1997), care, informal, spune că oricare două metode aproximative de optimizare sunt la fel de bune, în medie, peste toate problemele de optimizare, ceea ce poate fi citit şi „la fel de puțin bune”, pentru fiecare metodă există probleme pentru care metoda nu poate da soluții satisfăcătoare.

O coincidență

Înainte de a trece la calculul cu ADN, o precizare autobiografică. Eram la Graz, în Austria, în aprilie 1994, la o conferință, atunci când am intrat în posesia lucrării lui Tom Head, de la State University of New York din Binghamton, SUA, ulterior prieten şi colaborator, „Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviors”, apărută în *Bulletin of Mathematical Biology*, vol. 49, 1987, pp. 737-759. A fost o revelație. Mă găseam după aproape douăzeci de ani de lucru în teoria limbajelor formale şi intuiam acum un domeniu extrem de promițător de aplicații. E adevărat, ar fi trebuit să am această revelație mai demult, pentru că parcursesem la vremea ei lucrarea acad. Solomon Marcus „Linguistic structures and generative devices in molecular genetics”, din *Cahiers de Linguistique Thèorique et Appliquée*, vol. 11, 1974, pp. 77-104, dar probabil nu venise pe atunci vremea trecerii la bioinformatică, nu parcursesem cei douăzeci de ani de pregătire – pe care-i voi descrie pe scurt într-o secțiune ulterioară. Tom Head introduce în lucrarea sa o operație cu şiruri care formalizează operația de recombinare a moleculelor de ADN din genetică. O numeşte *splicing* şi aşa îi voi spune în continuare, pentru a o distinge de

recombinarea de la algoritmii genetici, cu care seamănă, fără a fi identică. Am imaginat chiar atunci, în vremea conferinței, un fel de gramatică bazată pe această operație, de fapt, pe o variantă mai simplă şi mai apropiată de operațiile cu şiruri decât operația lui Tom Head. Lucrarea a fost publicată în revista *Discrete Applied Mathematics* şi a consacrat versiunea de splicing pe care o propusesem. Peste puține săptămâni, eram la Leiden, în Olanda, unde am scris o lucrare împreună cu Grzegorz Rozenberg şi Arto Salomaa, cel de-al doilea, de la Turku, Finlanda, locul unde aveam să petrec apoi mai mulți ani, dedicați inițial calculului cu ADN şi apoi calculului membranar. Inspirat ca totdeauna, G. Rozenberg a titrat lucrarea noastră „Computing by splicing". Pentru că, începând de atunci, am numit *sisteme H* mecanismele de calcul bazate pe splicing, amintind astfel de cel care a introdus (inventat sau descoperit?...) operația cu pricina, i-am trimis lucrarea, în manuscris, lui Tom Head. Acesta a răspuns imediat, prin telefon, întrebându-ne excitat: ştiați că tocmai s-a efectuat un experiment reuşit de calcul cu ADN?! Nu, nu ştiam, a fost o coincidență – pe care o trec la coincidențe semnificative.

Primul calcul în eprubetă

Tom Head se referea la experimentul lui Leonard Adleman, publicat chiar în toamna lui 1994, în revista *Science*, nr. 226, nov. 1994, pp. 1021-1024: „Molecular computation of solutions to combinatorial problems". Despre posibilitatea de a folosi molecule de ADN pentru a calcula s-a speculat încă din anii 1970 (Ch. Bennett, M. Conrad, chiar şi R. Feynman, cu mult citata sa frază „există foarte mult loc acolo, jos", cu referire la fizică, dar extinsă afirmația şi la biologie). Adleman a confirmat această aşteptare, rezolvând în laborator problema dacă există un drum hamiltonian într-un graf (am amintit-o într-o secțiune anterioară). Problema este cunoscută ca fiind **NP**-completă, deci, printre cele mai grele probleme nefezabile, de complexitate exponențială (plecăm de la premisa că nu avem **P** = **NP**), dar Adleman o rezolvă într-un număr de paşi care este liniar în raport cu mărimea grafului. E adevărat, paşi biochimici, făcând uz de un paralelism masiv, ba chiar şi de nedeterminism, toate acestea posibile datorită caracteristicilor moleculelor de ADN şi biochimiei aferente.

Pe scurt, milioane de cópii ale unor secvențe simple de nucleotide, codificând nodurile şi arcele grafului, au fost puse împreună în soluție apoasă. Prin scăderea temperaturii, aceste secvențe au format molecule dublu catenare, corespunzătoare drumurilor în graf. Pentru că au fost folosite suficient de multe cópii ale secvențelor inițiale, cu mare probabilitate, au fost obținute *toate* drumurile din graf. Dintre acestea, au fost selectate cele care treceau prin toate nodurile, prin proceduri obişnuite de laborator: electroforeză pe gel pentru a separa moleculele de lungime potrivită lungimii drumurilor, apoi selectare prin denaturare şi amplificare prin PCR a drumurilor care treceau prin toate nodurile (deci, hamiltoniene).

Procedura presupune un număr de operațiuni biochimice liniar în raport cu numărul nodurilor grafului. Problema este **NP**-completă, deci realizarea este extraordinară – iar ecoul a fost pe potrivă. Deja în anul următor, 1995, s-a organizat la Princeton o întâlnire cu titlul „DNA Computing", care s-a transformat în conferință şi continuă şi astăzi. Da, dar...

Argumente pro şi contra

A fost un pas istoric, demonstrația că *se poate*. Pe un graf cu 7 noduri, însă, pentru care problema se poate rezolva printr-o simplă inspecție vizuală. Iar la începutul anilor '90, calculatoarele obişnuite puteau deja rezolva problema existenței dumurilor

hamiltoniene în grafuri cu câteva sute de noduri, suficient pentru aplicațiile practice curente (între timp, au progresat mult).

Iar soluția a fost obținută printr-o tranzacție spațiu-timp, numărul de molecule a fost exponențial de mare în raport cu numărul nodurilor. Juris Hartmanis, un clasic al informaticii, după ce şi-a exprimat entuziasmul (Hartmanis compară informatica cu fizica: a doua progresează în urma *experimentelor cruciale*, prima în urma *demonstrațiilor* că ceva se poate; Adleman a produs o asemenea demonstrație!), a calculat cantitatea de ADN necesar pentru a aplica procedura lui Adleman la un graf cu 200 de noduri şi a găsit că greutatea acestuia ar fi depăşit greutatea Pământului...

Cam aici este şi acum calculul cu ADN din punct de vedere practic. Numeroase experimente, dar tot pe „toy problems", multă teorie, priceperea în manevrarea moleculelor de ADN în laborator a progresat foarte mult, s-au obținut şi rezultate de interes pentru tehnologia generală de laborator (de pildă, o versiune îmbunătățită a reacției de polimerizare în lanț, PCR; unul dintre inventatori este matematician, Vincenzo Manca, amintit încă de la primele pagini ale acestui text), dar pe total domeniul s-a deplasat spre nanotehnologie, aplicațiile practice nu au apărut încă (dacă nu cumva există aplicații în criptografie, dar sunt clasificate).

Şi totuşi, lista posibilelor avantaje ale folosirii ADN-ului pentru a calcula este lungă: o foarte bună eficiență ca suport de date, la nivelul unui bit pe o nucleotidă; eficiență energetică; comportare paralelă şi nedeterministă, două visuri ale informaticii (cu mențiunea că nedeterminismul aduce şi probleme, de exemplu, posibile false soluții); manevrabilitate deosebită în laborator, stabilitate, reversibilitatea unor procese.

Minunata dublă elice

Molecula de ADN are caracteristici surprinzătoare la nivel informațional şi computațional. Să ne reamintim, formulat în termeni „sintactici": avem două şiruri de litere A, C, G, T, cele patru nucleotide, aşezate față în față, în perechi complementare Watson-Crick, totdeauna A făcând pereche cu T şi C cu G. Cele două şiruri sunt orientate diferit unul față de celălalt, orientarea fiind indicată de biochimişti prin notarea unui capăt de şir cu 3' şi a celuilalt cu 5'. Iar aici apare deja o surpriză matematico-informatică, semnalată de G. Rozenberg şi A. Salomaa în Raportul Tehnic 96-28 al Universității din Leiden, Olanda (octombrie 1996), „Watson-Crick complementarity, universal computations, and genetic engineering": structura moleculei de ADN „ascunde", codificată, puterea de calcul a maşinii Turing! Formularea este imprecisă, ea corespunde însă următoarei observații. Încă din 1980 s-a demonstrat (J. Engelfriet şi G. Rozenberg) că *orice limbaj ale cărui şiruri pot fi recunoscute de o maşină Turing, poate fi scris ca imaginea unui anume limbaj fixat, să-l notăm cu TS(0,1), printr-un traducător secvențial.*

Limbajul anterior este aşa-numitul „twin-shuffle" peste 0, 1 (de aici notația folosită): „shuffle" nu este altceva decât operația de amestecare a literelor a două cuvinte, fără a le schimba ordinea (exact ca la amestecarea cărților de joc din două pachete de culori diferite). Aici, amestecăm literele a două cuvinte „gemene", un şir de simboluri 0 şi 1 şi unul identic modulo schimbarea „culorii" acestor simboluri (le putem adăuga o bară deasupra sau semnul „prim" pentru a obține „şirul geamăn"). La rândul său, un traducător secvențial este cel mai simplu traducător, care are o memorie finită şi parcurge un şir de la stânga la dreapta. De remarcat că lucrăm cu patru simboluri, 0, 1 şi perechile lor, să zicem, 0' şi 1'. Exact atâtea nucleotide avem, patru. Să mai remarcăm şi că *TS(0,1)* este un limbaj fixat. Dându-se un limbaj oarecare, oricare ar fi el

(recunoscut de o maşină Turing dată), el se obţine plecând de la acelaşi unic *TS(0,1),* doar traducătorul depinde de limbaj.

Surpriza interesantă şi semnificativă este că limbajul *TS(0,1)* se poate obţine prin „citirea" moleculelor de ADN, în felul următor: mergem de-a lungul celor două şiruri complementare Watson-Crick, de la stânga la dreapta, avansând la întâmplare pe cele două şiruri şi asociind celor patru nucleotide A, C, G, T simboluri 0, 1, după regula: A = 0, G = 1, T = 0', C = 1'. Ceea ce obţinem, punând la un loc toate aceste citiri, pentru toate moleculele posibile de ADN, este exact *TS(0,1)!*

Prin urmare, orice limbaj care poate fi definit de o maşină Turing poate fi obţinut traducând aceste citiri ale moleculelor de ADN cu ajutorul celui mai simplu traducător, cel cu memorie finită. Traducătorul depinde de limbaj, el „extrage" din *TS(0,1)* rezultatul calculelor unei maşini Turing. Puterea este *acolo*, rămâne numai s-o punem în evidenţă. (Într-un anume sens, avem din nou cuplarea dintre un proces simplu, „citirea" moleculei de ADN, şi un observator de ordinul întâi, simplu, traducătorul secvenţial, ca în lucrările aminite mai devreme ale lui M. Cavaliere şi P. Leupold, cu rezultatul egalând nivelul maxim de calculabilitate, cel al maşinii Turing.)

Apar în acest context două chestiuni. Vorbeam, de exemplu, despre orientarea diferită a celor două catene ale moleculei de ADN, dar mai devreme le parcurgeam pe amândouă de la stânga la dreapta. Nicio problemă, citirea şirurilor duble poate să pornească din direcţii diferite şi rezultatul este acelaşi. Doi: natura este redundantă, sunt toate cele patru nucleotide (cele patru simboluri 0, 1, 0', 1') necesare pentru a acoperi, în sensul anterior, puterea de calcul a maşinii Turing? Nu, trei simboluri sunt suficiente – dar nu două!

Demonstraţii pentru toate aceste rezultate pot fi găsite în monografia (tradusă în japoneză, chineză şi rusă) Gh. Păun, G. Rozenberg, A. Salomaa, *DNA Computing. New Computing Paradigms*, Springer-Verlag, 1998.

Vorbind despre calcule şi redundanţă, să ne amintim că o mare parte din ADN este „reziduală", nu codifică gene şi nu ştim exact la ce foloseşte. Putem specula: dacă în celulă, la nivel genetic, se efectuează calcule (viruşii sunt şiruri de nucleotide, prin urmare şi recunoaşterea lor este o operaţie de analiză sintactică, deci un calcul), iar aceste calcule este de presupus că sunt complexe, de ce nu?, chiar la nivelul maşinii Turing, atunci avem nevoie de „spaţiu de lucru", de o „bandă" care în final rămâne în mare parte goală, rezultatul este aşezat la începutul ei (în cazul benzii maşinii Turing). ADN-ul aparent fără funcţie poate fi tocmai spaţiul de lucru pentru calcule complexe, pe care nu le înţelegem încă.

Calculând pe baza operaţiei de splicing

Adleman nu a folosit operaţia lui Tom Head în experimentul său, dar ingrediente biochimice specifice splicing-ului au fost utilizate în multe alte cazuri: enzime restrictive, care taie moleculele de ADN în contexte bine precizate, ligaze care leagă la loc moleculele, recombinarea pe baza „capetelor lipicioase" ale moleculelor cu cele două catene de lungimi diferite, deci cu nucleotide care nu-şi au perechea Watson-Crick în faţă.

Nu intru nici în amănunte biochimice şi nici în detalii matematice privind operaţia de splicing. Pe scurt, două molecule (reprezentate ca şiruri simple, pentru că nucleotidele de pe o catenă sunt identificate de perechile lor de pe cealaltă catenă) sunt tăiate în două bucăţi fiecare, la mijlocul unui context specificat printr-o pereche de subşiruri, iar fragmentele sunt recombinate încrucişat, obţinându-se două şiruri noi. Plecând de la o mulţime iniţială de şiruri şi aplicând această operaţie în mod repetat (în

raport cu o mulțime dată de contexte, deci de *reguli de splicing*), obținem un dispozitiv de calcul, un generator de limbaje, similar unei gramatici. Obținem un *sistem H*. Mare parte din monografia *DNA Computing. New Computing Paradigms* menționată mai devreme este dedicată studierii acestor sisteme: variante, putere generativă, proprietăți.

Atunci când se introduce un nou model de calcul, prima chestiune care trebuie clarificată privește puterea sa, în comparație cu etaloanele teoriei automatelor şi limbajelor formale – maşina Turing şi restricțiile sale, gramaticile Chomsky, sistemele Lindenmayer. Să notăm doar că cei doi „poli" ai calculabilității sunt puterea maşinii Turing, prin teza Turing-Church fiind nivelul maxim al calculabilității algoritmice, şi puterea automatului finit, nivelul minim. În termeni de gramatici şi limbaje, clasa maximală este cea a gramaticilor Chomsky generale şi a limbajelor recursiv enumerabile, iar cea minimală corespunde gramaticilor şi limbajelor regulate.

Sistemele H cu un număr finit de şiruri de plecare şi un număr finit de reguli de splicing nu generează decât limbaje regulate. Insuficient ca putere, mai ales că un „calculator" de acest nivel nu are proprietăți (convenabile) de universalitate, deci nu e programabil.

Interesant şi atrăgător este însă că, adăugând un minim control asupra operației de splicing, cu multe sugestii venind din zona gramaticilor cu restricții în derivare sau din biologie (exemplu: fiecare regulă de splicing are asociat un simbol-promotor şi ea nu se aplică decât şirurilor care conțin acest simbol; variantă – simbolul să nu apară, inhibitor), atunci obținem sisteme H echivalente cu maşina Turing. Demonstrația este constructivă, ca urmare, se „importă" astfel de la maşina Turing şi existența maşinii universale, aşadar, avem la dispoziție un sistem H universal, programabil.

Cu o mențiune: nu s-a putut realiza în laborator un asemenea „calculator" bazat pe splicing. Saltul de la cazul din natură, cu operația necontrolată (care se opreşte la puterea automatului finit), la cazul controlat, nu a putut fi făcut şi nici nu este clar dacă se va putea face curând. Realizarea calculatorului universal pe bază de splicing mai trebuie să aştepte...

Un detaliu important: funcționarea autonomă

Să nu uităm că un calculator universal, programabil, ar trebui să lucreze autonom, în sensul că odată un program pornit, calculatorul continuă fără intervenții din afară – cu totul altfel decât se întâmplă de obicei în experimentele de calcul cu ADN, unde operatorul uman controlează întregul proces. De pildă, în cazul experimentului din 1994, Adleman a fost, de fapt, „calculatorul", el a folosit moleculele de ADN doar ca suport de calcul, complexitatea de calcul s-a referit la paşii de laborator efectuați de biochimist, nu la paşii interni, la operațiile cu ADN, efectuate în paralel.

Există însă progrese promițătoare spre realizarea de calcule autonome, cuvântul-cheie, mult promovat în ultimii ani, fiind *auto-asamblare* (*self-assembly*). Realizări remarcabile în această direcție au fost obținute de Erik Winfree şi echipa sa de la Caltech, Pasadena, SUA, iar abordarea sa este de semnalat şi pentru că pleacă (reconfortant, printre altele, şi pentru discuția privind utilitatea matematicii), de la un capitol vechi al informaticii teoretice, calculul cu dominouri al lui Wang Hao, dezvoltat pe la începutul anilor '60 ai secolului trecut. Pe scurt, dominouri pătrate, cu laturile colorate (marcate), pot fi folosite pentru a calcula (prin alăturare, în aşa fel încât dominourile vecine să aibă laturile de aceeaşi culoare), simulând calculele unei maşini Turing. Obținem din nou un model de calcul care este universal.

Erik Winfree a realizat „dominouri" din ADN, cu laturile marcate cu secvențe potrivite de nucleotide, pe care le-a lăsat libere în soluție, pentru a se alătura conform

afinității Watson-Crick a nucleotidelor care „colorează" laturile. Abordarea a funcționat, experimentele au reuşit – dar totul a rămas, în termenii lui Hartmanis, tot la nivelul unui *demo*. Este important să subliniem că de data aceasta nu este vorba despre rezolvarea unei probleme anume, ca la Adleman şi la majoritatea experimentelor din literatura calculului cu ADN, ci de implementarea în laborator a unui model de calcul universal, autonom – de aceea, un *demo*, poate, mai important decât cel al lui Adleman (doar că Adleman a fost primul...).

Există şi alte încercări de a realiza în laborator „calculatoare" autonome. Semnalez doar simularea unui automat finit, un succes al unei echipe de la Weizmann-Rehovot şi Tehnion-Haifa, Israel: Y. Benenson, T. Paz-Elizur, R. Adar, E. Keinan, Z. Livneh, E. Shapiro, „Programmable and autonomous computing machine made of biomolecules", *Nature*, vol. 414, nov. 2001, pp. 430-434, cu mențiunea că era vorba despre un automat cu numai două stări. Tot numai un *demo*...

Ce înseamnă a calcula *în mod natural*?

Monografia *DNA Computing* are capitole dedicate şi altor moduri de a calcula inspirate din biochimia ADN-ului, de pildă, prin inserare şi ştergere de subşiruri (în contexte date), printr-un fel de „joc de domino" cu molecule de ADN care se cuplează pe baza complementarității Watson-Crick, diferit de modelul lui Wang Hao.

Splicing, inserare-ştergere, prelungire de şiruri. În calculul membranar întâlnim procesarea de multiseturi. Evoluția însăşi este bazată mai ales pe recombinare/splicing, mutațiile locale apar doar accidental. În contrast, calculatoarele existente şi modelele teoretice de calcul folosesc, aproape toate, operația de rescriere (*rewriting*). Se operează local, asupra unui şir oricât de lung. Acest lucru este adevărat pentru automate, gramatici, sisteme Post, algoritmi Markov. Toate aceste operații, şi rescrierea şi cele „naturale" (splicing-ul doar cu un control suplimentar), atât de diferite între ele, conduc la modele de calcul care au puterea maşinii Turing.

Întrebarea se impune: ce înseamnă a calcula *în mod natural*? Cu multe continuări: De ce informatica nu a considerat (decât sporadic) şi alte operații decât rescrierea? Se pot concepe calculatoare (electronice) folosind „operații naturale"? Folosind recombinarea de tip splicing, de pildă. Atunci când Hilbert a formulat întrebarea „ce este calculabil mecanic?", el avea în minte sisteme formale, în care substituția este o regulă de inferență centrală, iar Turing a propus un răspuns în acelaşi limbaj. Am fost oare astfel influențați, atunci când am proiectat calculatoarele, să gândim în aceiaşi termeni? Nu am citit pe undeva ca inginerii să fi spus că nu se pot imagina şi calculatoare bazate pe altfel de operații.

Rămâne întrebarea dacă astfel de calculatoare vor fi sau nu mai bune decât cele existente. Teoretic, vor avea aceeaşi putere, diferențele trebuie căutate pe alte coordonate: eficiență computațională, uşurință în utilizare, posibilități de învățare etc.

Spuneam mai devreme că sistemele H sunt fie de puterea automatelor finite, fie echivalente cu maşina Turing. Situații similare întâlnim şi în calculul membranar. Putem atunci afirma că clasele de automate şi gramatici intermediare între automatele finite şi maşina Turing, şi sunt multe astfel de clase studiate în informatica teoretică, nu sunt „naturale"? Într-un anume sens, acesta este cazul. De exemplu, limbajele independente de context au o definiție cu motivare matematico-lingvistică, în timp ce limbajele dependente de context au o definiție care ține de teoria complexității (referindu-se la spațiul de lucru necesar pentru a genera sau recunoaşte un şir).

Să trecem la celulă!

În ciuda acumulărilor teoretice, a numeroaselor experimente reuşite (totuşi, având de a face cu probleme de dimensiuni derizorii) şi a perfecţionării continue a tehnicilor de laborator, calculul cu ADN nu a confirmat entuziasmul de acum douăzeci de ani, de după anunţarea experimentului lui Adleman. Dacă nu cumva, spuneam, vor fi existând aplicaţii în criptografie despre care vom afla abia peste câteva decenii. Putem găsi amănunte care sprijină această bănuială. De pildă, la prima ediţie a conferinţei de DNA Computing, Princeton, 1995, una dintre comunicări (D. Boneh, C. Dunworth, R. Lipton: „Breaking DES using a molecular computer") descria modul în care se putea sparge Data Encryption Standard, DES, sistemul folosit de administraţia americană, folosind ADN, în patru luni. Anul următor, subiectul a fost discutat de o echipă din care făcea parte şi Adleman, iar experimentul propus putea sparge DES chiar în cinci zile, dacă operaţiunile de laborator ar fi fost robotizate. A mai fost prezentată o lucrare de acest gen şi în 1997, an în care DES a fost spart şi cu calculatoare electronice, drept care sistemul a fost retras.

Oricum, la câţiva ani de la experimentul lui Adleman devenise clar că nu se poate merge prea departe, mai era nevoie de încă o idee inovatoare, de încă o „străpungere", pentru a face un pas esenţial spre aplicaţii (spre o „killer-app", cum spun americanii), iar una dintre „explicaţii" se referea la faptul că ADN-ul nu se comportă *in vitro* la fel de bine (robust, predictibil) ca *in vivo*. Ideea apare de la sine: să mergem spre celulă!

La nivel personal, momentul coincidea şi cu scrierea monografiei *DNA Computing,* un fapt care s-a repetat aproape sistematic în primele două decenii ale carierei mele de cercetător: după aproximativ cinci ani de lucru într-o ramură a informaticii, am adunat, singur sau în colaborare, rezultatele într-o monografie, apoi am trecut la alt subiect – rămânând, totuşi, în cadrul informaticii teoretice, mai ales al limbajelor formale şi automatelor. Lipsă de tenacitate sau curiozitate? Poate din amândouă câte ceva, dar o combinaţie norocoasă: toate capitolele de informatică teoretică pe care le-am frecventat înainte de a trece la calculul membranar au fost folosite, uneori decisiv, în acest ultim domeniu – cu care am întrerupt definitiv tradiţia schimbării la fiecare cinci ani: după 17 ani dedicaţi aproape exclusiv calculului membranar, chiar dacă am scris, ca de obicei, o monografie după aproximativ cinci ani de la prima lucrare, nu a apărut încă niciun semn de ofilire a interesului.

Fascinanta celulă

Celula este cu adevărat fascinantă, mirabilă în sens blagian, pentru un matematician-informatician. Presupun că şi pentru biolog. Cea mai mică entitate despre care toată lumea este de acord că e *vie*. Subiectul este netrivial: la Institutul de la Santa Fe, New Mexico, SUA, de studiu al complexităţii, a fost iniţiat pe la mijlocul anilor 1980 un nou domeniu de cercetare, sub numele de *viaţă artificială*, o extindere a inteligenţei artificiale, care dorea studierea vieţii în sine, simularea ei pe suporturi nebiologice, pe calculator, modelarea matematică. S-a început, desigur, cu căutarea unei definiţii a ceea ce numim intuitiv *viaţă*, dar nu s-a ajuns prea departe: orice definiţie fie lăsa pe dinafară ceva viu, fie ne asigura, de exemplu, că viruşii de calculator sunt vii (au „metabolism", autoreproducere etc.). Să ne reamintim că Erwin Schrödinger are o întreagă carte al cărui titlu se întreabă *Ce este viaţa?,* tradusă la Editura Politică, în 1980.

Celula trece însă acest test. Ea este o extraordinar de mică „uzină", cu o structură internă complexă, ingenioasă şi eficientă, în care evoluează un număr enorm

de agenți, de la ioni la macromolecule precum cea de ADN, şi unde procese informaționale au loc la tot pasul. Unele celule trăiesc pe cont propriu (nu spun izolate), în organisme unicelulare, altele formează țesuturi, organe, organisme.

Este o discuție de interes aceea privind rolul celulelor în a face posibilă viața însăşi. Citez doar cartea de referință B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter, *Molecular Biology of the Cell*, 4th ed. Garland Science, New York, 2002, articolul lui Jesper Hoffmeyer: „Surfaces inside surfaces. On the origin of agency and life", *Cybernetics and Human Knowing*, vol. 5, 1998, pp. 33-42, important pentru cele ce urmează fiindcă ne propune sloganul „viața înseamnă suprafețe înăuntrul altor suprafețe", cu referire la membranele care dau structura interioară a celulelor, şi închei cu un paragraf din S. Kauffman, *At Home in the Universe*, Oxford University Press, 1995: „Secretul vieții, izvorul reproducerii, nu este de găsit în frumusețea împerecherii Watson-Crick, ci în realizările închiderii catalitice colective".

Adaug şi o sugestivă ecuație-slogan, pe care acad. Solomon Marcus a lansat-o la unul dintre primele workshopuri de calcul membranar, cel de la Curtea de Argeş, din 2002: *Life = DNA software + membrane hardware*.

Membrana. De la biologie la informatică

Am ajuns astfel la un ingredient fundamental – membrana. Se poate vorbi foarte mult despre aceasta, au făcut-o biologii, au făcut-o biosemioticienii. Celula însăşi există pentru că este separată de mediul înconjurător printr-o membrană. Nu numai metaforic vorbind, orice entitate autonomă există pentru că este delimitată de o „membrană", fie ea şi virtuală, de lumea din jur.

Celula (eucariotă) are un număr de membrane şi în interior: cea care înconjoară nucleul, complicatul aparat Golgi, vezicule, mitocondrii. Din punct de vedere computațional, principalul rol al acestor membrane este acela de a delimita „reactoare protejate", compartimente în care are loc o biochimie specifică. Mai sunt şi alte caracteristici-funcții ale membranei biologice care au importanță pentru calculul membranar: în membrane sunt plasate canale proteice care permit comunicarea selectivă între compartimente; pe membrane sunt fixate enzime care controlează multe dintre procesele biochimice din jur; membranele sunt utile şi pentru a crea spații cât mai reduse, în care moleculele aflate în soluție să aibă şansa de a se întâlni pentru a reacționa. Se spune că atunci când un compartiment este prea mare pentru ca biochimia locală să fie eficientă, natura creează membrane, pentru a obține „reactoare" suficient de mici (astfel încât, prin mişcare browniană, moleculele să se întâlnească suficient de frecvent, putând astfel reacționa) şi pentru a crea noi „suprafețe de reacție".

Subliniez că privesc celula, structura şi procesele care au loc în ea cu ochelarii informaticianului-matematician, ignorând o mulțime de detalii biochimice (de pildă, structura însăşi a membranelor) şi interpretându-le pe cele selectate conform obiectivului acestui demers: definirea unui model de calcul.

Să detaliem puțin, începând cu rolul esențial în comunicare. Dacă, în celula biologică sau în modelul pe care-l vom defini, compartimentele delimitate de membrane ar evolua separat, nu am avea un „reactor", ci un număr de „reactoare" alăturate, dar lucrând independent. Membranele asigură însă integrarea acestora. Moleculele polarizate sau de dimensiuni mari nu pot trece printre moleculele (fosfolipidice, cu un „cap" polarizat şi „picioruşe" hidrofobe, ale) membranelor, dar pot trece dintr-o parte în cealaltă prin canalele proteice. Această trecere este selectivă, uneori făcându-se împotriva gradientului, de la o concentrație mai mică la una mai mare. Cazul foarte interesant este cel al trecerii simultane printr-un canal proteic a două sau mai multor

molecule: moleculele respective nu pot trece separat, dar o pot face împreună, fie în aceeaşi direcţie (*simport*), fie o moleculă intrând în compartimentul respectiv şi cealaltă ieşind (*antiport*). Un capitol important şi extins al calculului membranar este bazat pe aceste operaţii, interesul venind şi din specificul acestui proces: nu există rescriere, ci doar transport de obiecte peste graniţele definite de membrane, nu există ştergere, există doar comunicare. *Calcul prin comunicare*. Să ne reamintim în acest context întrebarea *ce înseamnă a calcula în mod natural.*

Despre procesele informaţionale care au loc în celulă, eventual cu implicarea membranelor, citim în multe locuri.

„Multe proteine din celulele vii par a avea ca primă funcţie a lor transferul şi procesarea de informaţie, mai degrabă decât transformarea intermediarilor metabolici sau construirea de structuri celulare. Aceste proteine sunt legate funcţional prin mecanisme alosterice sau de altă natură în 'circuite' biochimice care efectuează o varietate de sarcini computaţionale simple, incluzând amplificarea, integrarea şi stocarea de informaţie."

Acesta este chiar rezumatul articolului lui D. Bray, „Protein molecules as computational elements in living cells", apărut în *Nature*, vol. 376, iulie 1995, pp. 307-312. La rândul lor, S.R. Hameroff, J.D. Dayhoff, R. Lahoz-Beltra, A.V. Samsonovich, S. Rasmussen, într-un articol din *Computer,* nov. 1992, pp. 30-39, privesc citoscheletul ca automat, în timp ce W.R. Loewenstein, în *The Touchstone of Life. Molecular Information, Cell Communication, and the Foundations of Life,* Oxford University Press, 1999, construieşte o întreagă teorie plecând de la aspectele informaţionale ale vieţii celulei. Despre biosemiotica membranei a vorbit în multe locuri Jesper Hoffmeyer, pe care l-am amintit şi mai devreme. Citez aici doar lucrarea sa „Semiosis and living membranes", *Seminário Avançado de Comunicaçao e Semiótica. Biossemiótica e Semiótica Cognitiva*, Sao Paolo, Brasil, 1998, pp. 9-19.

În context, putem aminti rolul esenţial al apei în viaţa celulei, implicit, procesele de comunicare transmembranară a acesteia, prin canale dedicate, *acvaporinele* în descoperirea cărora colegul Gheorghe Benga are contribuţii de pionierat.

O paranteză terminologico-istorică

Înainte de a trece la descrierea rapidă a calculului membranar, câteva precizări.

Mai întâi, despre numele domeniului. I-am zis *membrane computing*, plecând de la rolul membranei în viaţa celulei şi în arhitectura modelului, dar alegerea nu a fost cea mai potrivită. În primul rând, nu se traduce fericit în limba română. „Calcul cu membrane" e prea concret, „calculabilitate membranară" este lung şi nefiresc, am ales aici „calcul membranar", ca un compromis între cele două. „Calcul celular" era, probabil, alegerea cea mai „comercială", dar era un pic prea cuprinzător.

Apoi, numele modelelor: în primele lucrări, vorbeam despre „membrane systems", dar foarte curând cei care le-au preluat le-au numit „P systems", continuând şirul altor sisteme cu nume din informatică (Post, Lindenmayer şi H sunt cele mai apropiate), la început creându-mi un oarecare disconfort public, la conferinţe, de exemplu, dar iniţiala s-a autonomizat rapid, devenindu-mi complet neutră. Interesant este că există deja lucrări care folosesc sintagma (uneori chiar în titlu) fără a mai cita lucrări ale subsemnatului. Va fi, desigur, un mare succes dacă ea va deveni preponderent „folclorică"...

Domeniul a crescut rapid şi e încă activ, după şaisprezece ani de la pornirea la drum. M-am întrebat uneori care sunt explicaţiile – evident, şi pentru a vedea ce aş putea face pentru a-i sprijini creşterea. Au concurat multe lucruri la interesul pentru

calculul membranar: contextul favorabil („moda" calculului natural despre care am mai vorbit); momentul potrivit, pe de o parte, în raport cu calculul cu ADN (care este, într-un anume sens, înglobat în şi generalizat de calculul membranar), pe de altă parte, în raport cu informatica teoretică în general şi teoria limbajelor formale în particular.

Aici sunt mai multe de spus. După patru decenii de la introducerea gramaticilor Chomsky, teoria limbajelor formale se „clasicizase", retrăgându-se din fluxul principal al cercetărilor (aproape cu totul în SUA), chiar dacă încă există conferinţe specializate (de pildă, despre automate finite şi aplicaţiile acestora) sau mai generale (DLT – Developments in Language Theory). Calculul membranar a apărut ca o continuare şi extensie a acesteia: obiectul principal de studiu nu mai sunt şirurile de simboluri şi limbajele, ci (anticipez) multiseturile de simboluri şi mulţimile de multiseturi. Şiruri, fără a lua în seamă ordinea simbolurilor, mai tehnic vorbind, şiruri „văzute" prin funcţia lui Parikh, cea care spune câte apariţii are fiecare simbol într-un şir, exact multiplicitatea de la multiseturi. Urmarea a fost că un număr semnificativ de cercetători în teoria limbajelor formale au fost interesaţi de noul domeniu. Printre ei, încă de la început, nume mari, precum Arto Salomaa (Finlanda) şi Grzegorz Rozenberg (Olanda), Oscar H. Ibarra (SUA), Sheng Yu (Canada), Kamala Krithivasan (India), Takashi Yokomori (Japonia), Mario J. Pérez-Jiménez (Spania), plus cercetători foarte activi din generaţia mea, precum Jürgen Dassow (Germania), Erzsébet Csuhaj-Varjú (Ungaria), Jozef Kelemen (Cehia), Rudolf Freund (Austria), Gheorghe Marian şi Gabriel Ciobanu (România), Yurii Rogozhin (Republica Moldova, dispărut de curând), Linqiang Pan (China), mulţi dintre aceştia coagulând în jurul lor grupuri de cercetare dedicate calculului membranar.

Oarecum surprizător a fost numărul rapid crescător al doctoranzilor – azi, doctorilor – care au susţinut teze în acest domeniu. Sunt peste 50 la ora aceasta, nu-i menţionez decât pe primii, Shankara Narayanan Krishna (India) şi Claudio Zandron (Italia), cu tezele susţinute deja în 2001, respectiv, 2002. Din vara aceasta, C. Zandron este preşedintele comitetului de iniţiativă (Steering Committee) al domeniului.

O bogată informaţie despre calculul membranar se poate găsi la pagina internet de la adresa http://ppage.psystems.eu, organizată la Viena (succesoarea unei pagini care a funcţionat mai mulţi ani la Milano, la adresa http://psystems.disco.unimib.it).

A contat mult, desigur, „sociologia" domeniului. S-a format curând o *comunitate,* lucru foarte important, nu numai în ştiinţă, ci în cultură, în general. Au contribuit la aceasta seniorii amintiţi mai devreme, conferinţele anuale (încă din anul 2000, cu primele trei ediţii organizate la Curtea de Argeş, unde a revenit şi ediţia a zecea şi unde intenţionez să o organizez şi pe a douăzecea) şi, aş sublinia cu deosebire, o întâlnire de un tip inedit, pe care am organizat-o pentru prima dată la Tarragona, Spania, în 2003, şi de atunci, anual, la Sevilla, tot în Spania. Pentru că trebuia să poarte un nume, am numit-o „Brainstorming Week on Membrane Computing". O săptămână în care cei interesaţi de calculul membranar lucrează împreună, departe de grijile curente, sarcini didactice sau administrative. O idee utilă a fost colecţionarea de probleme deschise, circulate printre participanţi cu ceva vreme înainte de întâlnire şi abordate apoi, în colaborare, la Sevilla. Întâlniri extrem de productive – la pagina web a domeniului pot fi găsite volumele anuale, cu lucrările scrise sau doar începute la brainstorming.

Extrem de util a fost, desigur, internetul. Primul articol, „Computing with membranes", a aşteptat un an şi ceva până să apară în *Journal of Computer and System Sciences* (vol. 61, 2000, pp. 108-143), dar, pentru că eram la Turku în toamna lui 1998, am circulat lucrarea prin internet, sub forma unui raport intern al TUCS, Turku Center

for Computer Science (Report No. 208, 1998, www.tucs.fi). Până în 2000, când a apărut articolul tipărit, se scriseseră deja câteva zeci de lucrări, făcând posibilă organizarea primei întâlniri dedicate subiectului, cea de la Curtea de Argeş.

O privire rapidă asupra calculului membranar

Să nu uităm: vrem să pornim de la celulă şi să construim un model de calcul. Rezultatul (cel propus în toamna anului 1998) este ceva de genul următor. Privim celula şi abstractizăm până nu mai vedem decât structura de *membrane*, aranjate ierarhic, definind *compartimente* în care sunt plasate *multiseturi* de *obiecte* (folosesc un termen generic, abstract, eliberat de concreteţea biochimică); aceste obiecte evoluează conform unor *reacţii*. Un multiset este o mulţime cu multiplicităţi asociate elementelor, deci poate fi descris de un şir; de exemplu, *aabcab* descrie multisetul care conţine 3 cópii ale lui *a*, două ale lui *b* şi una a lui *c*. Toate permutările şirului *aabcab* descriu acelaşi multiset. Reacţiile sunt descrise de reguli de „rescriere a multiseturilor", de forma $u \rightarrow v$, unde u şi v sunt şiruri care identifică multiseturi. Iniţial (la începutul unui calcul), în compartimentele sistemului nostru avem multiseturi date de obiecte. Regulile de evoluţie sunt puse la lucru – aplicându-se, precum reacţiile biochimice, în paralel, simultan tuturor obiectelor care pot evolua – şi astfel multiseturile se schimbă. Aplicarea unei reguli ca mai devreme presupune „consumarea" obiectelor din *u* şi introducerea obiectelor din *v*. De remarcat că şi obiectele şi regulile sunt localizate, plasate în compartimente, regulile dintr-un compartiment se aplică numai obiectelor din acel compartiment. Unele obiecte pot şi să treacă prin membrane. Continuăm până ce (ca la maşina Turing) nu mai putem aplica nicio regulă, calculul se opreşte. În acel moment, „citim" rezultatul calculului, de pildă, sub forma numărului de obiecte dintr-un compartiment specificat dinainte.

Procesare de multiseturi (de simboluri), în paralel, în compartimentele definite de o structură ierarhică de membrane – iată descrierea scurtă a unui „P system". O gramatică distribuită, lucrând cu multiseturi – iată legătura directă cu teoria limbajelor formale.

Iar „şantierul" care începe de aici pare fără sfârşit.

Mai întâi, pot fi introduse un număr enorm de clase de sisteme, motivate matematic, informatic, biologic sau dinspre aplicaţii.

Din punctul de vedere al matematicii, modelele trebuie să fie minimaliste, să conţină minimul necesar de ingrediente. Pentru informatică, un model de calcul este bine să fie cât mai puternic, dacă se poate, echivalent cu maşina Turing, şi eficient, dacă se poate, să rezolve probleme **NP**-complete în timp polinomial.

Biologia şi aplicaţiile furnizează o lungă listă de variante, începând de la modul de aranjare a membranelor (ierarhic, ca într-o celulă, sau în nodurile unui graf arbitrar, ca în ţesuturi sau alte populaţii de celule), la tipul obiectelor (simboluri, ca mai devreme, şiruri sau alte structuri de date, mai complexe, cum ar fi grafuri sau array-uri bidimensionale), tipul regulilor de evoluţie, strategia de aplicare a acestora, modul de definire a rezultatului unui calcul.

Am menţionat mai devreme regulile de rescriere a multiseturilor. Ele pot fi *arbitrare, necooperative* (cu multisetul stâng format dintr-un singur obiect, ceea ce corespunde gramaticilor Chomsky independente de context) sau, caz intermediar, *catalitice* (de forma $ca \rightarrow cv$, unde *c* este un catalizator care asistă obiectul *a* în transformarea lui în obiectele din multisetul *v*). Vin apoi regulile de *simport* şi de *antiport,* care doar mută obiecte dintr-un compartiment în altul (exemplu: regula *(u, out; v, in),* asociată unei membrane, mută obiectele indicate de *u* din membrană în

compartimentul imediat superior şi pe cele indicate de *v* în sens invers), sau regulile care modifică membranele însele. Foarte importante sunt regulile de *divizare* a membranelor, care măresc, exponenţial chiar, numărul de membrane din sistem. Multe alte reguli au fost investigate (de pildă, cu control asupra aplicării lor – cu promotori sau inhibitori), dar nu le mai menţionez, am depăşi prea mult cadrul informal al prezentării de faţă.

Atunci când obiectele din compartimente sunt şiruri, ele evoluează prin operaţii specifice şirurilor: rescriere, inserare şi ştergere de subşiruri, sau, pentru a face modelul şi mai unitar biologic, operaţia de splicing, de la calculul cu ADN.

O situaţie interesantă este cea în care în sistem lucrăm cu obiecte-simbol, deci cu numere, dar rezultatul este „citit" în afara sistemului, sub forma şirului obiectelor care părăsesc sistemul. A se observa diferenţa calitativă dintre structura de date din interior, multisetul, şi cea din exterior, şirul, care poartă şi informaţie poziţională.

Aplicaţiile, la rândul lor, au nevoie de o cu totul altă strategie de construire a modelelor – deloc minimalistă, ci adecvată realităţii modelate, şi unde nu puterea de calcul este de interes, ci evoluţia în timp a sistemului. Voi reveni la aplicaţii.

Peste această mică junglă de modele se suprapune programul de cercetare sugerat de informatica clasică: putere de calcul, forme normale, complexitate descriptivă, complexitate de calcul, programe de simulare etc.

Clase de rezultate (şi probleme)

Evident, nu voi relua teoreme precise, ci voi menţiona doar cele două clase principale de rezultate şi stilul lor.

Universalitate computaţională: majoritatea claselor de sisteme P sunt echivalente cu maşina Turing, sunt computaţional complete şi, prin demonstraţii, pentru că acestea sunt constructive, împrumută şi proprietatea de universalitate în sensul lui Turing. Adesea, acest lucru se obţine pentru sisteme de o formă redusă, particulară, cu un număr mic de membrane. De pildă, sisteme P de tip celulă, cu două membrane, cu reguli catalitice (deci nu de forma generală) pot calcula tot ce poate calcula maşina Turing.

Important: sunt suficienţi numai *doi* catalizatori. Este o problemă deschisă de peste un deceniu dacă sistemele cu un singur catalizator sunt universale. Conjectura este că răspunsul este negativ, dar demonstraţia întârzie să apară. Acesta este unul dintre cele mai interesante tipuri de probleme (multe deschise încă) în calculul membranar: identificarea graniţei dintre universal şi sub-universal.

Eficienţă: clasele de sisteme P care pot creşte (exponenţial) numărul de membrane pot rezolva în timp polinomial probleme **NP**-complete. Ideea este generarea în timp polinomial a unui spaţiu de lucru exponenţial şi folosirea lui, în paralel, pentru examinarea soluţiilor posibile ale unei probleme. Divizarea de membrane ajută, la fel crearea de membrane, la fel şi alte operaţii. Ca în cazul lui Adleman, avem din nou o tranzacţionare spaţiu-timp, cu menţiunea că aici spaţiul nu este furnizat dinainte, ci este construit de-a lungul derulării calculului, prin „mitoză" sau prin alte operaţii biologice, „realiste".

Există şi în această zonă probleme deschise privind graniţa dintre eficient şi neeficient, dar mai greu de formulat în cuvinte.

Interesant este însă un fapt oarecum neaşteptat. Cu reguli de forma $a \to aa$, aplicate în paralel, putem produce un număr exponenţial de cópii ale lui *a* într-un număr liniar de paşi. (În *n* paşi, obţinem 2^n cópii ale lui *a*.) Totuşi, un asemenea spaţiu de lucru exponenţial nu ajută pentru a rezolva în timp polinomial probleme de complexitate mare

– este ceea ce ne spune aşa-numita *teoremă Milano*, din teza de doctorat a lui Claudio Zandron. Dacă, însă, aceste obiecte sunt localizate, plasate într-un număr exponenţial de membrane, atunci lucrurile se schimbă. Altfel spus, nu numai dimensiunea spaţiului de lucru contează, ci şi structura acestuia, posibilitatea de a aplica reguli diferite în locuri diferite. O diferenţă subtilă, care nu ştiu să fi fost sesizată şi până acum, în alte contexte.

Pentru detalii, trimit la monografia *Membrane Computing. An Introduction*, publicată la Springer-Verlag în 2002 şi tradusă recent în chinezeşte, şi, mai ales, la *Oxford Handbook of Membrane Computing*, editat împreună cu G. Rozenberg şi A. Salomaa, la Oxford University Press, în 2010.

Semnificaţii pentru informatică şi pentru biologie

Un model de calcul de puterea maşinii Turing este un lucru bun, un calculator de acest nivel este universal nu numai în sens intuitiv, ci şi programabil. În plus, avem un calculator distribuit, paralel, cu un mare grad de nedeterminism controlat în diferite moduri inspirate din biologie.

Să observăm însă asemănările şi diferenţele dintre un program uzual de calculator, un set de instrucţiuni ale unei maşini Turing şi un set de reguli de evoluţie dintr-un sistem P. În limbajele de programare, programele sunt formate din instrucţiuni precis înlănţuite, eventual etichetate şi apelate prin intermediul etichetelor. La maşina Turing, secvenţa de instrucţiuni de aplicat este determinată de stările maşinii şi de conţinutul benzii. În cazul celulei, reacţiile sunt potenţiale, mulţimea lor complet nestructurată, iar aplicarea lor depinde de moleculele disponibile. Regulile de evoluţie aşteaptă datele cărora să se aplice, au o comportare concurenţială în raport cu acestea.

Diferenţele sunt vizibile şi ele ne sugerează din nou întrebarea *cum se calculează în mod natural,* adăugându-i acum întrebarea dacă se poate lucra cu programe de forma unor mulţimi complet nestructurate de instrucţiuni.

La prima vedere, probabil că reacţia biologului la rezultate de genul echivalenţei cu maşina Turing este ridicarea din umeri. Alt domeniu, alt limbaj, altă *carte*. Şi totuşi: dacă celula este atât de puternică din punct de vedere computational, atunci, conform unei teoreme vechi, a lui Rice („orice problemă netrivială – care are şi instanţe cu răspuns afirmativ şi instanţe cu răspuns negativ – asupra unui model de calcul echivalent cu maşina Turing este nedecidabilă algoritmic"), nicio întrebare netrivială asupra celulei nu poate fi rezolvată algoritmic, printr-un program. Iar biologul formulează zilnic asemenea întrebări: Care este evoluţia în timp a unei celule, a unei culturi de celule, a unui organism? Există o substanţă care se acumulează peste o anumită limită, într-un anume compartiment? Dacă adăugăm un multiset de molecule (un medicament), se îmbunătăţeşte starea unui organ (din puncte de vedere specificate)? Şi aşa mai departe. Dacă un model al celulei ar fi decidabil, am putea afla răspunsul la asemenea întrebări examinând (algoritmic) modelul, la o stare dată, iniţială. Pentru că acest lucru nu este posibil (nu se poate face în principiu, nu numai că nu putem noi, acum, aici), ne rămâne experimentul de laborator (costisitor şi de durată), experimentul pe calculator (ieftin, rapid, dar a cărui relevanţă depinde de calitatea modelului) şi, teoretic, abordarea nealgoritmică, ad-hoc.

Paragraful dinainte poate fi văzut şi ca o pledoarie pentru biologie de a învăţa limbaje noi, în particular, informatică teoretică, având astfel posibilitatea de a ridica probleme şi de a găsi soluţii care în limbajul dinainte nu apăruseră, poate nici nu se puteau formula. Acesta ar fi un pas esenţial spre infobiologie.

Trei probleme informatice inedite

În ceea ce priveşte semnificaţiile pentru informatică, să semnalăm un aspect remarcabil: calculul natural în general, cel membranar în particular, ridică probleme teoretice care nu au fost avute în vedere în informatica tradiţională. Iată trei dintre acestea, toate trei ţinând de teoria complexităţii.

Începând cu Adleman, majoritatea experimentelor de calcul cu ADN au plecat de la o instanţă a unei probleme şi au construit un „calculator" asociat instanţei. Teoria complexităţii calculului nu permite aşa ceva, ci pretinde soluţii *uniforme*, programe care pleacă de la problema ca atare şi care, primind o instanţă ca intrare, o rezolvă. Ideea este că în timpul programării se poate deja lucra la rezolvare, pretinzându-se apoi că soluţia a fost găsită mult mai rapid decât este cazul. De aceea, şi în soluţiile uniforme se limitează timpul alocat programării. Să-l limităm atunci şi în cazul în care se pleacă de la o instanţă a problemei, pentru a nu putea trişa nici aici. Problema relaţiei dintre soluţii uniforme şi semi-uniforme (cu timpul de programare limitat) încă nu este complet rezolvată, în ciuda importanţei pentru calculul natural. Pentru calculul membranar s-au obţinut o serie de rezultate – a se vedea, în special, lucrări recente ale lui Damien Woods (Caltech, SUA), Niall Murphy (Microsoft Research, Cambridge, UK), Mario J. Pérez-Jiménez (Universitatea Sevilla, Spania).

Apoi, în calculul cu ADN şi la fel în multe modele de calcul membranar, parte dintre paşii unui calcul sunt nedeterminişti, dar în final experimentul/calculul ne furnizează un rezultat univoc. Ideea este de a organiza calculul în aşa fel încât el să fie *confluent,* cu două variante: fie sistemul să evolueze nedeterminist, dar să „conveargă" spre o configuraţie unică, apoi să se comporte determinist, fie să „conveargă logic", indiferent cum evoluează, să dea acelaşi rezultat. Iarăşi, teoria complexităţii nu ia în seamă o asemenea comportare, intermediară între determinism şi nedeterminism.

În sfârşit, biologia prezintă situaţii în care resurse extinse stau în aşteptarea unor solicitări care activează porţiunea necesară din resursă. Cazurile creierului şi ficatului, din care folosim în fiecare moment doar o parte dintre celule, sunt cele mai la îndemână. Putem atunci imagina „calculatoare" – de pildă, sisteme neurale – cu un număr arbitrar de mare de neuroni, dar care nu conţin decât o cantitate limitată de informaţie; după ce introducem o problemă în sistem, acesta activează numărul necesar de neuroni pentru a o rezolva. O teorie adecvată acestei strategii nu există. Cum trebuie să fie reţeaua de neuroni pentru a „conţine o cantitate limitată de informaţie", cum trebuie definită şi măsurată această informaţie, când un sistem cu resurse pre-calculate este acceptabil/onest, nu ascunde soluţia problemei în structura sa?

Calculul natural nu numai că face necesară îmbunătăţirea unor rezultate vechi din informatică, dar motivează şi dezvoltări noi, care nu apăruseră până acum.

Despre tehnicile folosite în calculul membranar

Pentru a sublinia încă odată relaţiile dintre ramuri aparent depărtate ale informaticii teoretice, unitatea acesteia, şi faptul că în calculul membranar, în calculul natural în general, sunt valorificate foarte multe tehnici şi rezultate anterioare din informatică, amintesc câteva episoade din experienţa personală.

În prima demonstraţie de universalitate a sistemelor P am folosit rezultatul lui Yuri Matijasevich, invocat şi mai devreme, de caracterizare a mulţimilor de numere calculate de o maşină Turing ca soluţii de ecuaţii diofantice. Am realizat apoi că o demonstraţie ceva mai simplă se poate obţine pornind de la caracterizarea aceloraşi mulţimi de numere cu ajutorul gramaticilor matriciale. Lucrarea a fost publicată în această formă. În context, a apărut necesitatea îmbunătăţirii unor rezultate vechi din

acest domeniu. După o vreme, şi gramaticile matriciale au fost înlocuite în demonstrații, anume cu maşinile cu regiştri, studiate încă din anii ’60.

O tehnică şi mai veche mi-a fost utilă în prima demonstrație de universalitate a sistemelor H, anume modul de funcționare a sistemelor Post, introduse pe la începutul anilor 1940. Transpus la operația de splicing, acesta a dus la tehnica numită *roteşte-şi-simulează,* aproape standard pentru sisteme H şi variante ale acestora.

De gramatici matriciale m-am ocupat din primii ani de cercetare şi am încheiat cu o monografie (publicată în 1981, rămasă în limba română), extinsă la o carte în colaborare cu Jürgen Dassow, de la Magdeburg, Germania (apărută la Springer-Verlag, în 1989), dedicată tuturor restricțiilor în derivare. La fel s-a întâmplat cu alte domenii, care au fost utile din plin calculului membranar; gramaticile contextuale Marcus şi sistemele de gramatici sunt cele mai importante.

În matematică şi în informatica teoretică nu se poate spune apriori dacă şi când un subiect sau un rezultat va fi util...

Spiking neural P (SNP) systems

Merită o semnalare separată o clasă de modele inspirate din modul de funcționare a creierului, introdusă ceva mai târziu (M. Ionescu, Gh. Păun, T. Yokomori: „Spiking neural P systems”, *Fundamenta Informaticae*, vol. 71, 2006, pp. 279-308), dar care, se pare, va ajunge la implementări hardware utile informaticii mai repede decât alte clase de sisteme P (detalii despre acest lucru pot fi găsite în lucrarea „The stochastic loss of spikes in spiking neural P systems: Design and implementation of reliable arithmetic circuits”, de Zihan Xu, Matteo Cavaliere, Pei An, Sarma Vrudhula, Yu Cao, în curs de apariție în *Fundamenta Informaticae*).

În câteva cuvinte, este vorba despre „neuroni” legați prin „sinapse”, de-a lungul cărora circulă impulsuri electrice, produse în neuroni de reguli specifice. La fel ca în cazul neuronilor reali (vezi, de exemplu, W. Maass: „Networks of spiking neurons: The third generation of neural network models”, *Neural Networks*, vol. 10, 1997, pp. 1659-1671), comunicarea între neuroni este asigurată de impulsuri electrice identice, pentru care frecvența este relevantă, codificând informație. Altfel spus, abstractizând, importantă este distanța în timp între impulsuri. La fiecare moment, axonii sunt un fel de „coduri de bare”, secvențe de 0 şi 1 care se deplasează de la un neuron la altul. Evident, în model sunt ignorate multe detalii neurobiologice, dar, chiar şi la acest nivel reducționist, putem formula unele întrebări/sugestii privind relevanța pentru informatică. Într-un anumit sens, sistemele SNP folosesc *timpul ca suport de informație.* Distanța între două evenimente, două impulsuri aici, codifică un număr. Se poate construi un calculator cu o asemenea „memorie”? Consemnăm întrebarea ca speculație doar – provocatoare însă la nivel teoretic.

Un rezultat ce merită amintit se referă la căutarea de sisteme SNP care să fie universale în sensul lui Turing, adică să poată fi programate în aşa fel încât să simuleze oricare alt sistem SNP. Din echivalența cu maşina Turing, rezultă imediat că asemenea sisteme există, problema de interes priveşte numărul de neuroni dintr-un astfel de „creier universal”, apt să simuleze orice calcul posibil în orice sistem particular. Iar acest număr nu este deloc mare. În lucrarea „Small universal spiking neural P systems”, *BioSystems,* vol. 90, 2007, pp. 48-60, de Andrei Păun şi Gh. Păun, sunt folosiți 50 – 80 de neuroni, depinzând de tipul de reguli de producere a impulsurilor electrice, dar aceste numere au fost ulterior micşorate. În termeni gazetăreşti, se poate spune că „există creiere universale computațional formate din numai câteva zeci de neuroni”. Putem trage fie concluzia că un model de calcul de tipul sistemelor SNP este foarte puternic, de

fapt, că neuronii din aceste sisteme sunt *prea* puternici, fie că nivelul Turing de calculabilitate nu este prea cuprinzător – fie ambele concluzii. Evident, creierul uman nu funcționează ca o maşină Turing – chiar dacă paradigma computațională este utilă, la un anumit nivel, în modelarea funcționării sale.

Despre implementări

Calculul cu ADN a debutat prin introducerea operației de splicing în 1987, dar despre posibilitatea de a calcula folosind molecule de ADN s-a vorbit cu un deceniu şi ceva mai înainte, iar domeniul a devenit popular după experimentul lui Adleman din 1994. S-a creat astfel un precedent, drept care întrebarea dacă există implementări ale sistemelor P este firească şi frecventă. Se înțelege, este vorba despre implementări pe un substrat biologic. Răspunsul este negativ. Au existat încercări, dar încă nu s-a raportat un experiment reuşit.

Un asemenea experiment a fost proiectat în grupul profesorului Ehud Keinan, de la Institutul Tehnion din Haifa, unde am petrecut o săptămână în 2006, tocmai cu acest scop. Trebuiau depăşite două probleme principale, corelate: identificarea unui sistem P care să facă posibilă reproducerea în laborator a funcționării lui şi, desigur, găsirea tehnicilor biochimice de implementare. Nu am intenționat rezolvarea unei probleme **NP**-complete, nu am întrezărit una abordabilă, ci am căutat un sistem a cărui comportare să fie ilustrativă pentru calculul membranar (compartimente, multiseturi, procesare paralelă) şi ne-am oprit la generarea de numere din şirul lui Fibonacci. Realizarea în laborator părea doar o chestiune de timp – şi bani, pentru achiziționarea aparaturii de laborator şi a... moleculelor de ADN. Membranele se doreau simulate prin microcamere într-o instalație reconfigurabilă de laborator, iar obiectele urmau să fie molecule de ADN.

Primele încercări nu au reuşit, apoi... sociologia ştiinței şi-a spus iar cuvântul: cele două doctorande care aveau ca sarcină experimentul s-au mutat în SUA. E adevărat, între timp a apărut un patent SUA pe numele lui Ehud Keinan (un cercetător de mare anvergură, specialist în chimie şi biologie deopotrivă), de implementare a unui sistem P, dar folosind o altă tehnică, bazată pe „trei lichide nemiscibile" plasate într-un spațiu comun. Din câte ştiu, e o „implementare de principiu", nu s-a făcut cunoscut niciun experiment reuşit.

Întrebarea care se pune este dacă un asemenea experiment ar aduce ceva cu adevărat util din punctul de vedere al aplicațiilor. Repetând o vorbă a lui Benjamin Franklin, „este imposibil de spus ce va ajunge un nou-născut", dar, având în minte situația de la calculul cu ADN, probabil că nu va fi decât tot un *demo*, la nivelul unor calcule simple.

Cu totul alta este situația implementărilor pe hardware electronic. Există multe implementări promițătoare pe hardware paralel (pe plăci NVIDIA, la Sevilla), pe hardware special proiectat pentru calcul membranar (la Madrid şi Adelaide, Australia), pe rețele de calculatoare, pe web chiar. Toate acestea reuşesc în mare măsură să prindă caracteristica esențială a sistemelor P, paralelismul. Având în vedere paralelismul, nu numesc implementări, ci simulări, cazurile în care se folosesc calculatoare obişnuite, secvențiale.

Pe de altă parte, şi simulările şi, cu atât mai mult, implementările sunt utile în aplicații.

Aplicaţii

Calculul membranar confirmă o observaţie făcută în multe alte situaţii: atunci când o teorie matematică, pornită de la un fragment precis de realitate, se dezvoltă suficient de mult la nivel abstract, teoretic, sunt toate şansele ca ea să se aplice şi în domeniul care a inspirat-o, dar şi în alte locuri, unele depărtate, la prima vedere, de cel de unde teoria a plecat (dar cu o structură de adâncime comună). Este, foarte pregnant, cazul de aici.

Revenirea la celulă era firească. Biologia duce lipsă de modele, celula nu este uşor de modelat. S-a şi afirmat că, după încheierea citirii genomului uman, principala provocare pentru bioinformatică este modelarea şi simularea celulei (M. Tomita: „Whole-cell simulation: A grand challenge of the 21st century", *Trends in Biotechnology*, vol. 19, 2001, pp. 205-210). Spuneam, multe modele folosite în biologie sunt bazate pe ecuaţii diferenţiale. În unele locuri ele sunt adecvate, în altele nu. Ecuaţiile ţin de matematica continuului, se potrivesc populaţiilor foarte mari de molecule, uniform distribuite. În celulă, multe molecule se găsesc în numere mici, drept care aproximarea finitului prin infinit, pentru a aplica ecuaţii diferenţiale, duce la rezultate viciate. De aici, firească, necesitatea modelelor discrete, în particular, a sistemelor P, care mai au şi alte caracteristici atrăgătoare pentru biolog: vin din biologie, sunt deci uşor de înţeles, aspect ce nu trebuie deloc subestimat; sunt modele algoritmice, direct programabile, pentru a fi simulate pe calculator; pot fi uşor extinse (sunt *scalabile*), adăugarea de componente, de orice tip ar fi acestea, nu schimbă programul de simulare; comportarea este de tip emergent, nu poate fi prezisă pe baza componentelor.

Sunt numeroase aplicaţiile în biologie şi biomedicină. De la celula individuală s-a trecut la populaţii de celule (bacterii), s-a trecut apoi la... ecosisteme. Doar un titlu, pentru că este sugestiv: „Modeling ecosystems using P systems: The bearded vulture, a case study", de Mónica Cardona, M. Angels Colomer, Mario J. Pérez-Jiménez, Delfi Sanuy şi Antoni Margalida, ultimii doi fiind specialişti în ecologia vulturului bărbos şi protecţia animalelor, din Lleida, Spania. Evident, ecosistemul este o celulă metaforică, „moleculele" care interacţionează fiind vulturii, caprele sălbatice, lupii, vânătorii – toate acestea în cantităţi discrete, numere cunoscute, fără nicio şansă de a fi modelate cu instrumente ale matematicii continue. Alte ecosisteme studiate privesc urşii panda din China şi scoica dungată din lacurile de acumulare ale hidrocentralelor spaniole.

Aplicaţii plauzibile până aici. Mai puţin aşteptate sunt cele în grafica de calculator (dar avem precedentul sistemelor Lindenmayer), criptografie (în organizarea atacului asupra unor sisteme criptografice), optimizare aproximativă (calcul evolutiv distribuit, cu distribuirea organizată ca într-o celulă; numărul lucrărilor de acest gen este foarte mare, subiectul fiind popular în China, iar rezultatele sunt surprinzător de bune – cu menţiunea că faimoasa *no free lunch theorem* cenzurează şi aici entuziasmul), modelare economică (o extensie metaforică similară celei la ecosisteme), controlul roboţilor.

Aceste din urmă două arii de aplicaţii sunt parte a uneia potenţial mai ample, care se bazează pe folosirea aşa-numitelor *sisteme P numerice*, unde, într-un cadru „celular" evoluează nu molecule, ci variabile numerice, pe baza unor *programe* constituite dintr-o *funcţie de producţie* şi un *protocol de repartiţie.* Inspiraţia vine din economie (Gh. Păun, Radu Păun: „Membrane computing and economics: Numerical P systems", *Fundamenta Informaticae*, vol. 73, 2006, pp. 213-227). Sistemele de acest tip calculează funcţii de mai multe variabile, în paralel, iar acest calcul este extrem de

eficient, de aceea este de aşteptat ca această clasă exotică de sisteme P să-şi afle şi alte aplicații.

Detalii despre aplicații pot fi găsite la pagina web a domeniului, în *Handbook,* precum şi în volumele colective *Applications of Membrane Computing* (editat de G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez) şi *Applications of Membrane Computing in Systems and Synthetic Biology* (editat de P. Frisco, M. Gheorghe, M.J. Pérez-Jiménez), ambele apărute la Springer-Verlag, în 2006, respectiv, 2014.

Îndoieli, dificultăți, neîmpliniri

În momente festive, cum este şi cel de față, sau cu ocazia periodicelor raportări, este neuzual, poate şi nepotrivit, să vorbim şi despre neîmpliniri şi momente de cumpănă – chiar dacă acest lucru poate fi instructiv pentru ascultător/cititor şi util pentru domeniu.

Pe de altă parte, ezitările şi îndoielile îl însoțesc continuu pe cercetător. Aş putea, de pildă, face o listă lungă de momente în care aşteptările au fost de un fel, rezultatele de un alt fel.

Începând cu rezultatele matematice. De exemplu, la început nu am crezut că sistemele P catalitice sunt universale, cu atât mai mult cele cu numai doi catalizatori. La fel, am tot sperat în găsirea de clase de sisteme pentru care numărul de membrane să inducă o ierarhie infinită. În schimb, mai mereu universalitatea este obținută cu una sau două membrane. O singură membrană înseamnă nicio structurare a sistemului, o arhitectură trivială. Sigur, putem vedea partea plină a paharului: procesarea (catalitică) a multiseturilor este suficient de puternică pentru a simula maşina Turing.

Pentru că am în minte situația de la calculul cu ADN, nu trec la neîmpliniri lipsa unei implementări biologice decât pentru valoarea publicitară a unui asemenea eveniment, dar sunt încă în aşteptarea unei implementări pe hardware dedicat care să fie de interes practic „comercial". Este nevoie de aşa ceva şi, cred, este şi posibil. De pildă, acum câțiva ani, o echipă de biologi şi informaticieni din Nothingham, Sheffield şi Sevilla au încercat să simuleze pe calculator comunicarea între bacterii, modelând aşa-numitul *quorum sensing.* Simulatoarele puteau lucra cu sute de bacterii, biologii doreau să treacă la populații de mii de bacterii. Mă aştept ca implementările pe hardware paralel (pe plăci NVIDIA, de exemplu) să acceadă curând la acest ordin de mărime cerut de biologi.

Apropo de aplicații, chiar dacă acestea nu mă interesau la început, a devenit la un moment dat evident că domeniul nu poate trece de un anumit nivel de dezvoltare şi notorietate fără aplicații „reale". Aplicații apăreau, dar de tip *postdicție*, nu *predicție.* Scenariul „clasic" este următorul: se ia un fenomen biologic discutat într-o lucrare sau într-o carte, se formalizează ca sistem P, se scrie un program (sau se foloseşte unul aflat în circulație – acum, avem şi un limbaj de programare specializat, *P-lingua*, realizat la Universitatea din Sevilla), se fac experimente cu datele din carte şi, dacă rezultatele sunt similare celor obținute în laborator sau folosind alte modele, ne bucurăm. Postdicție, nimic nou pentru biologie, doar creşterea încrederii în noul model. Pentru a trece de acest stadiu este nevoie de un biolog, care să vină cu o problemă de cercetare, cu ipoteze care trebuie testate, şi care să facă echipă cu informaticianul în aplicarea calculului membranar. La rândul său, informaticianul trebuie să vină cu modele suficient de versatile şi cu programe suficient de eficiente, pentru a face față complexității biologice. După şaisprezece ani de calcul membranar, bibliografia aplicațiilor de tip predicție este consistentă – a se vedea referințele din secțiunea anterioară – chiar dacă este încă nevoie de biologi care să vină spre informatician, eventual să învețe calcul membranar, sau

măcar să învețe să folosească instrumentele software pe care informaticianul le-a realizat deja.

Spuneam mai devreme că m-a preocupat formarea unei comunități – inițial intuitiv, apoi conştient, era un mod de a stabiliza domeniul în fața „fluctuației cadrelor", a dinamicii grupurilor. E un aspect care pare exterior, dar nu trebuie deloc subestimat efectul psiho-sociologiei asupra ştiinței, în special în cazul ramurilor tinere. Un grup care se sparge poate însemna un grup în minus (depinde unde ajung membrii lui, dacă îşi continuă sau nu activitatea de cercetare) sau apariția unor grupuri noi, multiplicate, în locuri inedite. Am asistat la ambele tipuri de urmări. Din fericire, comunitatea de calcul membranar are la ora aceasta dimensiuni care-i dau o inerție confortabilă – dar care nu garantează că domeniul nu se va dizolva în infobiologie, ba chiar dimpotrivă, lucrează în acest sens...

La frontiera dinspre SF

Principala promisiune a calculului natural este eficientizarea folosirii calculatorului existent, împingerea frontierei fezabilității, furnizarea de soluții, fie ele şi aproximative, la probleme care nu se pot rezolva prin tehnici tradiționale. Calculul cu ADN a venit cu un scop mai ambițios, inedit: furnizarea unui nou tip de hardware, a unor „cipuri biologice", „cipuri ude", eficiente nu numai în termeni computaționali, ci şi în ceea ce priveşte consumul de energie, sau făcând plauzibile caracteristici deosebit de atrăgătoare, cum ar fi autorepararea, adaptarea, învățarea. Biologia poate sugera şi noi arhitecturi de calcul sau idei de realizare a altor dorințe ale informaticii, cum ar fi calculul paralel, nesincronizat, controlul proceselor distribuite, calculul reversibil etc.

Toate acestea sunt aşteptări oarecum standard, dar există şi altele care trimit spre ştiința de mâine, dacă nu direct spre science-fiction.

Una dintre direcții este cea care țintește spre *hipercalculabilitate* – aşa sunt numite cercetările care vizează „calcularea necalculabilului", trecerea dincolo de bariera Turing. Domeniul este bine dezvoltat, există vreo duzină şi ceva de idei care conduc la modele de calcul mai puternice decât maşina Turing – iar fizica nu interzice niciuna dintre acestea, ba chiar sugerează idei cu adevărat SF, cum ar fi folosirea unui timp interior modelului care să conțină bucle sau să fie bidimensional. E adevărat, Martin Davis le consideră pe toate trucuri, prin care puterea de calcul se introduce de la început, deghizată, în modelul despre care apoi se arată că trece dincolo de maşina Turing (de exemplu, sub forma unor numere reale, care pot codifica în şirul lor infinit de zecimale toate calculele posibile), dar există unele idei „mai realiste" decât altele.

Una este cea a *accelerării*, o idee discutată de mult, nu numai în informatică: B. Russell (1936), R. Blake (1926), H. Weyl (1927) şi-au imaginat procese care au nevoie de o unitate de timp (măsurată de un ceas exterior) pentru primul pas, de o jumătate de unitate de timp pentru al doilea şi aşa mai departe, la fiecare pas, pe jumătate ca la pasul dinainte. În acest fel, în două unități de timp (insist: exterior, măsurați de observator) sunt executați un număr infinit de paşi. O maşină Turing accelerată astfel poate rezolva *problema opririi*, deci este mai puternică decât maşinile Turing obişnuite.

Să ne amintim acum observația că natura creează membrane pentru a realiza reactoare mici, în care reacțiile să fie favorizate, datorită posibilităților ridicate de ciocnire a moleculelor. Prin urmare, *mai mic* înseamnă *mai repede*. Biochimia dintr-o membrană interioară este mai rapidă decât cea din membrana de deasupra. Să ducem speculația până la capăt şi să presupunem că „viața" dintr-o membrană este de două ori mai rapidă decât cea din membrana care o conține. Exact accelerarea de care vorbeam mai devreme. Se poate demonstra (C. Calude, Gh. Păun: „Bio-steps beyond Turing",

BioSystems, vol. 77, 2004, pp. 175-194) că, exact ca la maşina Turing accelerată, un sistem P accelerat poate decide problema opririi.

Hipercalculabilitatea poate părea doar un exerciţiu matematic, dar se estimează că trecerea dincolo de bariera Turing ar putea avea consecinţe mai importante decât găsirea unei demonstraţii, chiar eficiente, pentru egalitatea **P** = **NP**; a se vedea, de exemplu, B.J. Copeland: „Hypercomputation", *Minds and Machines*, vol. 12, 2002, pp. 461-502.

Să ne apropiem însă de laborator. Am amintit de implementarea unui automat finit, cu funcţionare autonomă. Un automat face analiza sintactică a unor şiruri. Genele sunt şiruri, viruşii sunt şiruri (de nucleotide). O speranţă a medicinii este să vindece boli prin editarea genelor, să elimine viruşii prin identificarea şi apoi tăierea lor în fragmente. O idee mult mai eficientă decât introducerea de medicamente în corp este construirea unei „maşinării" care să recunoască şi să editeze secvenţele dorite de nucleotide, fie ele gene sau viruşi, iar pentru asta e necesar un vector care să ducă editorul de gene la locul potrivit. Identificarea acestui loc poate fi asigurată de un automat, eventual finit, vectorul ca atare poate fi un fel de nano-cărăuş care poate fi construit tot din molecule de ADN. Pe total, un nano-robot multiplicat corespunzător, care să umble din celulă în celulă, vindecând ce este de vindecat. Un preproiect al acestui nano-robot a fost prezentat în 2004, de Y. Benenson, E. Shapiro, B. Gill, U. Ben-Dor, R. Adar („Molecular computer. A 'smart drug' in a test tube"), la cea de a zecea ediţie a Conferinţei DNA Computing, Milano, Italia. Era în mare măsură aceeaşi echipă care a implementat automatul finit autonom amintit într-o secţiune anterioară.

Mai sunt multe lucruri de pus la punct, posibilitatea de a avea corpul scanat încontinuu de un robot reparator de gene nu este deloc aproape. (Un asemenea robot ar putea avea şi sarcini malefice, ar putea fi o armă – se poate deschide aici o discuţie despre etica cercetării, există destule dezbateri de acest gen, inclusiv în bioinformatică. Despre *bioetică* vorbeşte şi Francis S. Collins, în *Limbajul lui Dumnezeu*, cartea citată mai la început.) Există însă numeroase nano-construcţii din ADN, „motoare", „roboţi" etc., nano-tehnologia pe bază de ADN este spectaculos de dezvoltată. A se vedea, de exemplu, J.H. Reif, T.H. LaBean, S. Sahu, H. Yan, P. Yin: „Design, simulation, and experimental demonstration of self-assembled DNA nanostructures and motors", *Proceedings of the Workshop on Unconventional Programming Paradigms, UPP04*, Le Mont Saint-Michel, septembrie 2004, Springer, 2005.

Merită amintită aici şi o observaţie făcută de Jana Horáková şi Jozef Kelemen în „Capek, Turing, von Neumann, and the 20th century evolution of the concept of machine", din *Proceedings of the International Conference in Memoriam John von Neumann,* Budapest Polytechnic, 2003, pp. 121-135, privind evoluţia calculatoarelor, oarecum în paralel cu evoluţia ideii de robot, de la organic la electromecanic, apoi la electronic şi revenirea plauzibilă la organic.

Alte speculaţii? Fără limite, adesea plecând de la fapte cu suport ştiinţific solid. Spre latura extremă, ar putea fi menţionat Frank Tipler, cu controversata sa viaţă eternă, în termeni informaţionali – ceea ce nu este altceva decât viaţă artificială la scara universului (F. Tipler: *The Physics of Immortality*, Doubleday, New York, 1994). În orice caz, trebuie să fim conştienţi că mai toate acestea sunt planuri pentru mâine, dar formulate în limbajul de ieri, pentru a relua o spusă a lui Antoine de Saint-Exupéry. Progresele în bioinginerie ar putea să aducă surprize pe care nu ni le putem imagina acum.

Nu cumva sperăm prea mult?

Să coborâm însă cu picioarele pe pământ, la calculul natural aşa cum îl avem astăzi şi aşa cum este plauzibil să-l avem în viitorul apropiat, adoptând o poziţie lucidă, dacă nu chiar sceptică, contrară entuziasmului din secţiunea anterioară şi entuziasmului multor autori. (Nu mă refer şi la jurnalişti, care folosesc prea des cuvinte mari atunci când este vorba despre bioinformatică.)

Pentru a promova o ramură ştiinţifică tânără, entuziasmul este de înţeles şi este util – dar calculul natural nu mai este deloc tânăr. Să opunem, deci, optimismului de până acum, o poziţie mai realistă, pornind de la diferenţele, numeroase şi semnificative, între informatică şi biologie, de la dificultăţile de a implementa bio-idei în informatică şi calcule în celule: scopul vieţii este viaţa, nu calculul, noi, informaticienii, vedem calcule pretutindeni şi încercăm să le folosim pentru noi; într-un anume sens, viaţa are timp şi resurse nelimitate, îşi permite să experimenteze repetat, să renunţe la rezultatele nepotrivite – toate acestea sunt greu de extins la calculatoare, fie ele şi bazate pe biomolecule. La fel, viaţa are un mare grad de redundanţă, de nedeterminism. Apoi, procesele biologice au un grad mare de complexitate, mai mult, par a folosi mai ales matematica aproximărilor, probabilităţi, mulţimi vagi, care sunt mai greu de prins într-un model de calcul, cu atât mai greu de implementat.

Şi mai important: poate că visăm prea mult şi la nivel teoretic. În primul rând, tranzacţionarea spaţiu-timp nu poate redefini clasele de complexitate, cel mult măreşte spaţiul fezabilităţii (a se revedea remarca lui Hartmanis cu privire la experimentul lui Adleman).

Există, apoi, o teoremă a lui Michael Conrad („The price of programmability", în volumul *The Universal Turing Machine: A Half-Century Survey*, R. Herken, ed., Kammerer and Unverzagt, Hamburg, 1988, pp. 285-307) care ne spune că trei caracteristici dorite ale unui calculator, *programabilitatea* (universalitatea), *eficienţa* şi *evolvabilitatea* (capacitatea de adaptare şi învăţare), sunt contradictorii, nu există un calculator care să le posede pe toate trei în acelaşi timp. Putem interpreta acest rezultat ca o *no free lunch theorem* generală pentru calculul natural.

O teoremă similară de limitare a „ceea ce se poate face în principiu" este teorema lui Robin Gandy, doctorand şi colaborator al lui Turing, care îi oferă argumente matematice generale lui Martin Davis: hipercalculabilitatea este un lucru extrem de dificil de obţinut (a se vedea, de exemplu, articolul lui Gandy „Church's thesis and principles for mechanisms", în volumul *The Kleene Symposium*, J. Barwise et al., eds., North-Holland, Amsterdam, 1980, pp. 123-148). Gandy a dorit să elibereze teza Turing-Church de orice tentă antropică (în înţelegerea lui Turing, teza spunea că „tot ce poate fi calculat de o fiinţă umană abstractă care lucrează algoritmic poate fi calculat de o maşină Turing") şi, pentru aceasta, a definit o noţiune generală de „maşină", descrisă de patru proprietăţi formulate matematic, pe care orice „calculator", fizic sau teoretic, ar trebui să le aibă. El a demonstrat apoi că orice maşină cu aceste patru proprietăţi poate fi simulată de o maşină Turing.

Să trecem de la informatică la aplicaţii în biologie şi să începem tot prin a sublinia că există limitări serioase şi în această privinţă. Sunt chiar convins că, dacă s-ar face liste cu dorinţele pe care le avem de la modele şi simulări (adecvare, acurateţe, eficienţă, inteligibilitate, programabilitate, scalabilitate şi multe altele), vor apărea teoreme de imposibilitate de genul teoremelor lui Arrow, Conrad, Gödel privind modelarea celulei înseşi – spre simularea căreia ne chema M. Tomita.

Toate-s vechi şi nouă toate...

Şi totuşi, există o preocupare tot mai vizibilă privind modelarea celulei, ba chiar s-a propus o direcție dedicată de cercetare, *biologia sistemică* (*systems biology*), cu numeroase articole programatice apărute în reviste de mare vizibilitate, precum *Science* şi *Nature*. Principalul promotor a fost H. Kitano („Systems biology: A brief overview", *Science*, vol. 295, martie 2002, pp. 1662-1664, „Computational systems biology", *Nature*, vol. 420, nov. 2002, pp. 206-210), care are în vedere un model general al celulei, care să fie simulat pe calculator şi apoi folosit, în relație şi cu alte instrumente informatice şi biologice, până la „transformarea biologiei şi medicinei într-o inginerie precisă". Scopul este important şi probabil fezabil pe termen mediu-lung, dar insistența cu care s-a vorbit despre „systems biology" ca despre o noutate l-a făcut pe Olaf Wolkenhauer să se întrebe încă din titlul articolului său din *Briefings in Bioinformatics* (vol. 2, 2001, pp. 258-270) dacă aceasta nu este cumva doar „reincarnarea aplicării teoriei sistemelor în biologie". Lucrarea aminteşte eforturile în acest sens din anii 1960, cu dezamăgirile apărute atunci, datorate, printre altele, limitelor calculatoarelor din acea vreme (dar şi limitelor biologiei: să ne amintim că modelul Singer-Nicolson al membranei ca „mozaic fluid" datează abia din 1972). Dar, în afară de puterea de calcul, poate că mai lipsea ceva, care probabil lipseşte şi acum, în informatică şi biologie deopotrivă. Ultimul paragraf din lucrarea lui Olaf Wolkenhauer îl invocă pe Mihailo Mesarovic, un clasic al teoriei sistemelor, care spunea, în 1968, că „în ciuda considerabilului interes şi a eforturilor, aplicarea teoriei sistemelor în biologie nu a răspuns aşteptărilor. (...) Unul dintre motivele principale pentru care există acest decalaj este că teoria sistemelor nu a abordat direct probleme de interes vital pentru biologie." Sfatul său pentru biologi, adaugă Olaf Wolkenhauer, este că un asemenea progres se poate obține numai printr-o mai directă şi puternică interacțiune cu cercetătorii în teoria sistemelor. „Un avans real în aplicațiile teoriei sistemelor în biologie va apărea numai atunci când biologii vor începe să *formuleze întrebări* care sunt bazate pe concepte sistemice şi nu atunci când aceste concepte sunt folosite pentru a reprezenta într-un mod nou fenomene care sunt deja explicate în termeni biofizici sau biochimici. (...) În acel moment nu vom avea doar o aplicare a principiilor inginereşti la probleme biologice, ci un domeniu nou, al biologiei sistemice, cu propria-i identitate." (M.D. Mesarovic: „System theory and biology – view of a theoretician", în *System Theory and Biology*, M.D. Mesarovic, ed., Springer, New York, 1968, pp. 59-87.)

Cuvintele lui Mesarovic pot fi luate ca motto al infobiologiei pentru care pledează întreg textul de față.

Transformarea biologiei şi medicinei într-o „inginerie precisă" poate fi pusă în legătură şi cu dificultățile curente de a înțelege ce este viața, materializate, printre altele, în limitele curente ale inteligenței artificiale şi vieții artificiale. Se spune, de exemplu, că, până acum, calculatoarele sunt remarcabile în AI, amplificarea inteligenței, dar nu la fel de spectaculoase în IA, inteligență artificială. Toate acestea sugerează, în termenii lui Rodney Brooks („The relationship between matter and life", *Nature*, vol. 409, ian. 2001, pp. 409-411) că „probabil ne lipseşte ceva fundamental şi greu de imaginat la ora aceasta în modelele noastre biologice". Calculatoarele sunt bune la a lucra cu numere, dar nu „la a modela sisteme vii, fie ele mai mici sau mai mari". Intuiția este că viața este mai mult decât biofizică şi biochimie, dar ce altceva este poate fi ceva *inimaginabil*, „care ne este invizibil la ora aceasta. Nu este complet imposibil că vom descoperi noi proprietăți ale biomoleculelor sau noi ingrediente". Un exemplu de asemenea „altceva" pot fi efectele cuantice din microtubulele celulelor nervoase, care, conform lui Penrose, „pot fi locul conştiinței la nivelul celulei" (citat de R. Brooks).

O opinie similară a fost exprimată de un alt clasic al inteligenței artificiale, John McCarthy („Problems and projection in CS for the next 49 years", *Journal of the ACM*, vol. 50, 2003, pp. 73-79): „Inteligența de nivel uman este o problemă științifică dificilă și probabil are nevoie de idei noi. Acestea sunt mai plauzibil să fie imaginate de o persoană de geniu decât ca parte a unui proiect guvernamental sau industrial."

În orice caz, progresele legate de colaborarea dintre informatică și biologie nu trebuie subestimate. Dacă o facem, ne asumăm un risc care nu a ocolit nume mari ale științei și culturii. Ca picanterie de final, să-l dăm ca exemplu pe Auguste Comte, cu două fraze hazardate care i se pun în seamă (datate ca fiind de pe la 1830): „Orice încercare de a folosi metode matematice în studiul chestiunilor biologice trebuie considerată profund irațională și contrară spiritului biologiei. Dacă matematica va avea vreodată un loc proeminent în biologie – o aberație care este, din fericire, aproape imposibilă – aceasta va prilejui o rapidă și generalizată degradare a acestei științe"... Slavă Domnului, nu a avut deloc dreptate – dar ne-au trebuit aproape două sute de ani pentru a ne convinge de acest lucru...

Încheiere (provizorie)

Sper că această descriere succintă a convins că drumul de la biologie la informatică și înapoi la biologie este fascinant intelectual și util ambelor științe. Câteva lucruri merită reținute: (i) în toată istoria sa, informatica a încercat să învețe din biologie, (ii) iar acest efort a adus beneficii importante informaticii și deopotrivă biologiei; (iii) progresele în această direcție nu pot fi supraestimate, (iv) dar, în general, este plauzibil că așteptăm prea multe (și prea repede) de la simbioza informatică-biologie, (v) pentru că ignorăm diferențele esențiale dintre cele două universuri, limitele inerente ale calculabilității și faptul că biologia nu este o știință matematizată, (vi) cu mențiunea că este posibil să avem nevoie de o nouă matematică pentru a modela și simula viața și inteligența; în sfârșit, (vii) fie-mi permis să anticipez o nouă vârstă a biologiei, dincolo de bioinformatica și de calculul natural de azi, și să propun un termen care să o numească, *infobiologie*.

Să ne dăm un răgaz de încă două decenii pentru a o vedea conturându-se?

Din punct de vedere intelectual, cei patruzeci de ani despre care a vorbit acest text i-am petrecut în preajma domnului academician Solomon Marcus, un „mare copac" care invalidează fraza cu care Brâncuși și-a motivat refuzul de a lucra cu Rodin: profesorul Solomon Marcus nu i-a umbrit niciodată pe mulții studenți și colaboratori pe care i-a avut și-i are în juru-i, ba dimpotrivă. Repet, pentru a întări: dimpotrivă. Depun mărturie în acest sens și-i închin cu recunoștință prezentul discurs de recepție, mulțumindu-i încă o dată.