Educaţia Matematică Vol. 4, Nr. 1 (2008), 15 - 31

De la predicţia salturilor condiţionate la o întrebare fundamentală: ce este aleatorul? [1]

Lucian N. Vinţan

Abstract

In some previous published research in Computer Architecture we discovered some very difficult to predict branch instructions, called unbiased branches that have a "random" dynamical behavior. First, we tried to improve their predictability by enlarging their prediction information (branch's history contexts).

Second, we tried to find new relevant information in order to accurately predict them. Despite of our efforts, these unbiased branches still restrict the ceiling of dynamic branch prediction and therefore accurately predicting unbiased branches remains an open problem. Starting from this technical challenge, we tried to understand in more depth what randomness, from a mathematical point of view is. Essentially a random sequence can't be generated through an algorithm (Turing Machine). Random is closely related with Kolmogorov's complexity concept, too. Based on our bibliographical research focused on understanding what a random string is, finally we proposed some random degrees metrics, like program's complexity, compression rate, entropy (energy) and HMM prediction's accuracy, that might be useful for characterizing strings of symbols and particularly, our unbiased branches' behavior.

[1] *Received 26 April 2008*

Accepted for publication (in revised form) 30 June 2008

2000 Mathematical Subject Classification: 60J80, 60J85, 97U99

1. Problema salturilor condiţionate impredictibile

În microprocesoarele de azi şi de mâine există necesitatea predicţiei instrucţiunilor de ramificaţie (salturi condiţionate, branches; generate spre exemplu în urma compilării unor construcţii de limbaj de tipul if-then-else, loop etc.). Nu intrăm aici în amănunte, detalii se pot găsi în [9]; problema este însă una de mare importanţă în arhitectura calculatoarelor. Predicţia acestora (prae + dicere, în lb. latină) înseamnă ca încă din faza de aducere a instrucţiunii din memorie (instruction fetch) să se anticipeze (în timp real al microprocesorului) 3 aspecte:

1. Dacă instrucţiunea adusă este sau nu este una de branch.

2. În caz că este este branch, dacă acest salt condiţionat se face sau nu se face (Taken/Not Taken).

3. Dacă saltul se face, care este adresa sa destinaţie (Target Address).

În acest scop s-au dezvoltat predictoare markoviene, neuronale, bayesiene, bazate pe arbori de decizie etc., simplificate pentru a putea fi implementate în hardware-ul microprocesoarelor. În dezvoltarea celor de tip neuronal chiar autorul acestui articol are o prioritate [7].

Pentru fiecare branch dinamic (în curs de procesare), predicţia se face pe baza unei informaţii binare de context, ataşate respectivului branch şi considerate relevante pentru comportamentul său (istoria locală a branch-ului, memorată pe un anumit număr de biţi, istoria globală, calea de program pe care s-a ajuns la saltul condiţionat curent - branch's path etc.) Ce am observat statistic? Că anumite branch-uri dinamice, într-un anumit context binar de apariţie, au un comportament nepolarizat, fiind atât Taken cât şi Not Taken în proporţii semnificative (unbiased branches le-am numit în [8, 3]). Mai precis, se consideră contextul de apariţie al unui branch dinamic ca fiind pe p biţi.

Fiecare branch static are asociate k contexte dinamice de apariţie ($k \leq 2^p$). Indexul de polarizare al unui branch apărut într-un context dat se

defineşte ca fiind:

$$P(S_i) = \max(f_0, f_1) = \begin{cases} f_0, & f_0 \geq 0,5 \\ f_1, & f_0 < 0,5 \end{cases} \quad (1)$$

unde

- $S = \{S_1, S_2, \ldots, S_k\}$ = mulţimea contextelor distincte apărute pentru toate instanţele branch-ului;

- k = numărul de contexte distincte, $k \leq 2^p$;

- $f_0 = \frac{T}{T+NT}$, $f_1 = \frac{NT}{T+NT}$, NT = numărul de instanţe Not_Taken corespunzătoare contextului S_i, T = numărul de instanţe Taken corespunzătoare contextului S_i, oricare ar fi $i = 1, 2, \ldots, k$ şi evident $f_0 + f_1 = 1$;

- $P(S_i) \in [0.5, 1]$, semnificând pentru minimul 0.5 nepolarizare maximă a contextului S_i(fully unbiased).

În plus, aceste salturi condiţionate au şi un comportament haotic, entropic, adică în contextul dat sunt amestecate (Taken/Not_Taken) într-un mod ce pare a fi relativ aleator, măsurabil prin formula (3).

În consecinţă sunt practic impredictibile (comportamentul lor nu poate fi învăţat) şi de acest aspect ne-am convins prin experienţe proprii. Cum am abordat problema înţelegerii şi predicţiei lor?

Prima idee a fost aceea de a mări lungimea (istoria) contextului binar asociat fiecărui branch. Dacă la un context pe 32 de biţi cca. 17% din branch-uri erau unbiased şi impredictibile prin algoritmii şi schemele utilizate de noi, la unul pe 64 de biţi au scăzut la cca. 4%, ceea ce reprezintă totuşi mult. De precizat că un context de predicţie de 64 de biţi nu este fezabil din punct de vedere practic, implicând complexităţi exponenţiale ale predictoarelor aferente, deci problema este în realitate şi mai deranjantă.

Totuşi, după cum am intuit, o istorie mai lungă le-a micşorat entropia. Aceste branch-uri unbiased au repercursiuni negative asupra performanţei globale a microprocesoarelor.

Am arătat că cele mai sofisticate predictoare ale momentului, unele preluate direct din simulatoarele software publicate în cadrul Campionat-

ului mondial de branch prediction organizat de compania Intel [1], obţin acurateţi foarte scăzute de predicţie pentru aceste branch-uri, în jur de 70% [3] (în timp ce un salt "normal" este prezis cu acurateţi de 97%-99%). Predictoarele utilizate erau hibride, fiind compuse din seturi de predictoare Markov de diferite ordine (Prediction by Partial Matching) şi din predictoare neuronale.

A 2-a idee a fost aceea de a căuta alte informaţii de context, relevante, care să le reduca entropia (haosul) şi deci să devină astfel mai predictibile. Altfel spus, am căutat acele informaţii relevante prin prisma cărora branch-urile unbiased să devină mai polarizate şi deci, mai predictibile. Încercările în acest sens au fost realmente interesante şi instructive, dar au eşuat din punct de vedere al eficienţei practice. S-a încercat inclusiv predicţia valorii (semnului) condiţiei de salt întrucât aceasta determină comportamentul saltului [3].

Aşadar, în cazul acestor branch-uri nepolarizate, comportamentul lor memorat ca o lungă secvenţă (zeci de milioane) de '0' (Not_Taken) şi de '1' (Taken), este impredictibil din punct de vedere al nevoilor noastre inginereşti. De ce oare? În fond, ele sunt generate prin compilarea unor programe cu acţiuni deterministe, nu aleatoare. Or fi aceste branch-uri "aleatoare" sau doar relativ impredictibile prin structurile şi informaţiile de context utilizate?

În continuare se prezintă cîteva reflecţii asupra aleatorismului, cu scopul practic de a sugera câteva metrici care să caracterizeze o secvenţă (binară) aleatoare. Aceste metrici ar putea justifica teoretic mai bine aleatorismul (impredictibilitatea) acestor salturi condiţionate.

Scopul este dificil întrucât, în abordarea problemei aleatorului pe matematicieni nu-i prea interesează aspectele particulare, inginereşti, în timp ce o metrică acceptabilă este imposibil de propus fără a înţelege, cel puţin în principiu, ce cred matematicienii despre conceptul de aleator.

2. Ce este o secvenţă aleatoare? Câteva reflecţii matematico-filosofice

Aşadar întrebarea mai generală care se pune este: în fond, ce înseamnă aleator? În particular, ce înseamnă un şir binar aleator? De multe ori, aleatorul se confundă cu impredictibilul.

Totuşi, care ar fi diferenţele specifice între aceste două noţiuni? Impredictibilă este o secvenţă care nu poate fi prezisă printr-o metodă (algoritm) de predicţie. Se poate da o definiţie matematică intrinsecă, ontologică, a unui şir aleator de simboluri? (deci care să nu facă referire la metodele de predicţie ci doar la proprietăţi intrinseci ale şirului respectiv.)

Poate genera rularea unui program determinist secvenţe "aleatoare", sau acest fapt trebuie respins apriori? (Un şir aleator însă, ar putea conţine subşiruri deterministe sau măcar corelate stohastic, deci predictibile.) Există oare fenomene cu adevărat aleatoare în lumea fizicii macroscopice, guvernată de legi şi de modele matematice deterministe? Dar în lumea ştiinţei şi ingineriei calculatoarelor? Nu cumva aleatorul şi impredictibilul apar în această lume macroscopică doar ca efect al lipsei de informaţii relevante a observatorului sau a dificultăţilor acestuia de a gestiona multitudinea informaţiilor? (Este, spre exemplu, incomparabil mai comodă previzionarea rezultatului aruncării unei monede prin modele stohastice decât prin calcule deterministe.)

Altfel spus, modelele stohastice nu sunt oarecum artificiale, justificabile doar ca instrumente comode si relativ eficiente de contracarare a incertitudinii?

O informaţie relevantă asupra generatorului de secvenţe haotice (în general, greu sau chiar imposibil de găsit), nu le-ar reduce "devălmăşia" astfel încât aceste secvenţe să devină (mai) predictibile? Dacă este aşa, oare merită să încercăm să le predicţionăm cu algoritmi mai puternici, de genul modelelor adaptive Markov cu legături ascunse, Hidden Markov Models - HMM [6, 2]?

Această ipoteză a autorului se bazează pe faptul că aceste modele stohastice cu legături ascunse, care consideră că stările modelului Markov observabil sunt generate de un alt model Markov ascuns, ar putea compensa necunoaşterea acelor informaţii relevante pentru predicţie prin chiar procesul stohastic ascuns, pe post de generator al stărilor observabile (cu o semantică neinteligibilă pentru observator).

Prin utilizarea acestor modele HMM aleatorul n-ar mai constitui o fatalitate ireductibilă, din cauza modelului stohastic ascuns care îl generează pe cel observabil. Astfel, acurateţea predicţiei unui şir de simboluri printr-un astfel de model puternic, n-ar putea constitui oare un grad de aleatorism al secvenţei, din punct de vedere practic? Să considerăm acum, spre exemplu, secvenţele binare pe 23 de biţi:

- 01010101010101010101010
- 01101010000010011110011

Prima secvenţă apare ca fiind una periodică iar a 2-a pare a fi una aleatoare, deşi ambele au aceeaşi probabilitate de apariţie, anume $1/2^{23}$. De ce oare a 2-a secvenţă pare aleatoare iar prima nu? Nu cumva este aceasta o impresie psihologică ţinând de structura vizual-cognitivă a omului? Mai ales că şi cea de a 2-a secvenţă este una deterministă, reprezentând primele zecimale ale numărului iraţional $\sqrt{2}$.

Existând deci un algoritm care o generează, desigur secvenţa nu este aleatoare. Dacă însă algoritmul generator ne este ascuns, secvenţa ne apare ca fiind aleatoare. Se pare că nu ne putem baza prea mult pe intuiţia noastră de a deosebi aleatorul de non-aleator.

Considerăm acum un şir binar finit (secvenţă) $X1, X2 \ldots, Xk$, probabilitatea de apariţie a lui '0' fiind $P(0) = a$, iar cea de apariţie a lui '1' fiind $P(1) = b$, unde $a + b = 1$. Fără specificarea acestor probabilităţi, definirea noţiunii de secvenţă binară aleatoare pare să nu aibă sens. Astfel spre exemplu, o secvenţă binară având de 7 ori mai mulţi '0' decât '1' nu ar putea fi considerată aleatoare dacă $a = b = 1/2$. În schimb, secvenţa ar

putea fi considerată aleatoare dacă $a = 7/8$ şi $b = 1/8$. Dacă numărul de apariţii ale lui '0' în şir este m (implicit numărul de apariţii ale lui '1' este $k - m$), atunci probabilitatea de apariţie a secvenţei $X1, X2, \ldots, Xk$ este:

$$P(X1X2 \ldots Xk) = \prod_{i=1}^{k} P(X_i) = a^m b^{k-m} \tag{2}$$

În particular dacă $a = b = 1/2$ atunci $P(X1X2...Xk) = 1/2^k$. Totuşi, prin prisma acestor consideraţii, o anumită secvenţă ar fi la fel de aleatoare ca oricare alta, fapt aflat în contradicţie cu intuiţia noastră despre aleator, după cum am mai arătat.

O măsură a aleatorismului unei secvenţe S de simboluri, un simbol oarecare aparţinând mulţimii $X = \{X1X2...Xk\}$, s-ar putea baza chiar pe entropia informaţională discretă a secvenţei S, anume

$$E(S) = -\sum_{i=1}^{k} P(X_i) \log P(X_i) \geq 0,$$

cu un maxim ($\log k$) în cazul simbolurilor echiprobabile în S. Evident, dacă logaritmul se scrie în baza 2, E(S) reprezintă numărul minim de biţi care pot codifica simbolurile secvenţei. Totuşi, măsura entropiei nu este suficientă întrucât se pot găsi secvenţe cu entropie ridicată dar care să fie generate în mod determinist. În [Vintan06], relativ la o secvenţă binară S, s-a definit indicatorul numit grad de amestecare, astfel:

$$\begin{cases} 0, \ n_t = 0 \\ \dfrac{n_t}{2 \cdot \min('0','1')}, \ n_t > 0 \end{cases} \tag{3}$$

unde:

- n_t = numărul de tranziţii binare ($0 \rightarrow 1$ sau $1 \rightarrow 0$) din secvenţa S;

- min('0','1') reprezintă minimul dintre numărul de apariţii al lui '0' respectiv al lui '1' în secvenţa S. Deci $2 \cdot \min('0','1')$ = numărul maxim de tranziţii binare posibile într-o secvenţă de lungimea lui S.

Se observă imediat că $D(S) \in [0,1]$, fiind maxim în cazul unor amestecări maximale ale simbolurilor '0' şi '1' în secvenţă. Credem că o măsură inginerească acceptabilă de definire a gradului de aleatorism aferent unei secvenţe binare S, ar putea fi dată de produsul $D(S)E(S) \in [0, \log_2 k]$. Desigur că în locul entropiei $E(S)$ s-ar putea utiliza şi alte metrici precum energia informaţională Gini-Onicescu $En(S) = \sum_{i=1}^{k} P^2(X_i) \in [1/k, 1]$ cu minimul atins în cazul simbolurilor echiprobabile în S, sau sinergia informaţională definită ca $Sinerg(S) = -\log En(S)$.

Solomon Marcus a arătat că întrebările despre semnificaţia conceptului de şir aleator îi framântă pe matematicieni de peste 400 de ani şi a făcut o scurtă prezentare a problemei [Marcus08]. Astfel, în secolul trecut (1909) s-au propus diferite modele matematice ale aleatorului, primul find cel propus de către matematicianul francez Emile Borel, sub forma de şir normal (el a avut în vedere secvenţe descriind numere reale, dar chestiunea se extinde la şiruri infinite arbitrare de simboluri, pe un anumit alfabet finit).

Aleatorul după Borel ar reveni la faptul că, pentru orice n, blocurile de n termeni din şir au aceeaşi probabilitate de apariţie în şir. În cazul şirurilor pe un alfabet de m simboluri respectiv al şirurilor binare, această probabilitate este egală cu 1/mn respectiv1/2n, fiind în concordanţă cu legea numerelor mari. Borel a arătat că aproape toate numerele reale, scrise în reprezentare zecimală infinită, sunt aleatoare conform definiţiei sale. Acest fapt este justificabil şi pe criterii de calculabilitate-Turing, după cum se arată în continuarea acestei lucrări.

Totuşi, remarcăm că această definiţie a lui Borel nu este constructivă (efectivă), în sensul permiterii generării de şiruri aleatoare. Orice algoritm de concatenare a pattern-urilor de lungime dată conduce la negarea caracterului aleator al şirului. Acest fapt conduce la ideea că se pot construi şiruri Borel-normale care nu sunt şi aleatorii. În acest sens este cunoscut un alt contraexemplu şi anume şirul obţinut prin concatenarea numerelor prime (numărul Copeland-Erdos): 2357111317192329...

Analog numărul lui Champernowne obţinut prin concatenarea numerelor naturale: 0123456789101112131415 16... S-ar putea da alte multe exemple în acest sens.

Au urmat alte modele matematice, mai restrictive, bazate pe complexitatea descrierii unei secvenţe de simboluri. A. N. Kolmogorov (cel care a axiomatizat teoria probabilităţilor în 1933) şi G. Chaitin au definit, în anii '60 ai secolului trecut, aleatorismul unei secvenţe finite de simboluri, cu referire la lungimea celui mai scurt program de calculator care descrie secvenţa respectivă. Dacă lungimea acestui program este (sensibil) egală cu lungimea secvenţei însăşi, atunci secvenţa este considerată aleatoare (deci nu există un program generator, mai scurt). Într-o formulare echivalentă, un şir aleator nu poate fi comprimat.

Deci, din punct de vedere practic, rata de compresie a unei secvenţe de simboluri ar putea constitui o măsură a aleatorismului acelei secvenţe.

Totuşi, un corolar al acestei definiţii ar conduce la ideea inacceptabilă că majoritatea şirurilor mai scurte decât lungimea programului minim de generare a lor, sunt aleatoare. Evident că lungimea celui mai scurt program (algoritm) care poate genera un anumit şir de simboluri x, numită şi complexitate Kolmogorov K(x) sau entropie algoritmică, depinde de limbajul formal de descriere considerat.

Se arată că dacă $K_1(x)$ şi $K_2(x)$ sunt complexităţile Kolmogorov asociate limbajelor de descriere L_1 respectiv L_2, atunci exist'a $c \in N$ astfel încât $Abs(K_1(x) - K_2(x))$, unde prin *Abs* s-a notat funcţia modul. Din păcate utilitatea practică este mică pentru că o teoremă arată că K(x) nu este o funcţie calculabilă (v. în continuare), altfel spus nu există un algoritm care să proceseze secvenţa x şi să genereze la ieşire $K(x)$ [11]. În general, se consideră calculatorul ca fiind modelat de o maşină Turing (v. în continuare).

Ideea de esenţă este că unei complexităţi reduse i se asociază ordinea în timp ce uneia ridicate i se asociază aleatorul (haosul). În general, matematicienii au preferat să abordeze problema considerând mulţimea tuturor

şirurilor binare infinite în locul unor secvenţe finite. S-a propus ca un şir infinit să fie declarat aleator dacă orice prefix al său (segment iniţial) este aleator în sensul Kolmogorov-Chaitin, însă s-a arătat că nici această definiţie nu rezistă. Conform lucrării [10], pentru şiruri de simboluri matematicienii cred că nu se poate defini decât cel mult gradul de aleatorism, diferenţa între aleator şi non-aleator fiind deci una vagă.

Astfel, spre exemplu, dacă un şir binar considerat aleator conţine m zerouri, nu există niciun motiv ca să nu considerăm la fel de aleator şirul obţinut prin adăugarea sau ştergerea unui '0' la/din şirul iniţial. Analog, credem că inserarea unui şir determinist într-unul aleator conduce tot la un şir aleator. În cartea [4], este discutată problema relaţiei complexe dintre ordine şi haos, explicându-se de ce nicio definiţie a aleatorului nu satisface pe deplin.

Între conceptul de aleator şi cel de algoritm respectiv de calculabilitate există o strânsă dependenţă. Un şir de simboluri generat printr-un algoritm (funcţie calculabilă) nu este aleator, şirul fiind predictibil prin însuşi algoritmul care îl generează (pe post de predictor).

Pentru adâncirea acestei intuiţii este necesară reamintirea conceptului fertil de maşină Turing, descris în 1936 de către matematicianul britanic Alan Turing. O maşină Turing este constituită dintr-o bandă infinită de celule egale şi dintr-un cap de citire/scriere pe post de automat finit. Fiecare celulă conţine un simbol $s \in S = 0, 1, blank$. Secvenţa de intrare x aparţine mulţimii tuturor secvenţelor binare finite (FB - Finite Binary) şi poate codifica, în diverse moduri, datele de intrare. Capul de citire-scriere citeşte o celulă (simbol) în fiecare pas. Acest cap poate fi într-o stare aparţinând mulţimii $Q = \{q_0 q_1 q_2 \dots q_f\}$, unde q_0 este starea iniţială (start) iar este q_f cea finală (stop). Funcţie de simbolul $s(t)$ citit curent şi de starea curentă $q(t)$, capul de citire-scriere scrie un nou simbol $s(t+1)$ în celula curentă, se mută o poziţie la stânga (L, Left) sau la dreapta (R, Right) şi tranzitează într-o nouă stare $q(t+1)$.

Aşadar fiecare pas este caracterizat de 5-uplul $TM_t(x) = \{q(t), s(t); s(t+$

$1), q(t+1), m\}$ unde$m \in M = \{L, R\}$. $TM_t(x)$ se mai numeşte şi instrucţiune, secvenţa tuturor instrucţiunilor constituind programul executat de maşină. Aşadar din punct de vedere formal, o maşină Turing este o funcţie $f : QxS \rightarrow QxSxM$. Mulţimea TM a tuturor maşinilor Turing este una numărabilă[2]. Acest fapt se justifică imediat întrucât se poate trece de la mulţimea maşinilor Turing cu k instrucţiuni (TM^k) la a celor cu (k+1) instrucţiuni (TM^{k+1}) oricare ar fi $k = 3, 4, 5...$ rezultând deci că mulţimea TM^k este numărabilă.

Evident că în cadrul unei mulţimi TM^k există un număr finit de maşini Turing, deci mulţimea TM este numărabilă. Dacă o maşină Turing ajunge în starea finală (converge), calculul se opreşte. (Altfel, calculul diverge, continuând în mod nedeterminat.) În urma acestui calcul, pentru oricare ar fi $x \in FB$ se generează secvenţa de ieşire $TM(x)$. Prin urmare fiecare maşină Turing defineşte o funcţie parţială[3] a mulţimii FB pe ea însăşi, semnificând faptul că, în fond, calculul (algoritmul) este echivalent cu o procesare de simboluri.

Spre exemplu, se poate construi o maşină Turing care să adune, să înmulţească etc. oricare M numere întregi, fiecare codificat binar pe N biţi. Aceste operaţii nu sunt altceva decât funcţiile parţiale amintite. Un rezultat important constă în demonstrarea faptului că există o infinitate de aşa numite maşini Turing universale (TU). O astfel de maşină TU poate simula orice maşină Turing dacă i se codifică la intrare setul de instrucţiuni al maşinii simulate TM urmat de intrarea x. Evident, la ieşirea TU se va genera TM(x).

Orice şir binar din FB poate reprezenta un număr natural prin intermediul unei funcţii de codificare $c : FB \rightarrow N$. Conform lui Sergio Volchan [Volchan02], o funcţie parţială $f : N \rightarrow N$ este "Turing-calculabilă" dacă oricare ar fi $n \in N$, exista $x \in FB$ cu $n = c(x)$ pentru care maşina se

[2]O mulţime este numărabilă dacă poate fi pusă în corespondenţă bijectivă cu mulţimea numerelor naturale N

[3]Dacă funcţia este definită pe mulţimea tuturor şirurilor binare, se zice totală

opreşte generând la final codificarea binară a lui n prin funcţia c, adică $f(n) = c(TM(x))$. Cum mulţimea TM este numărabilă rezultă că mulţimea funcţiilor parţiale Turing-calculabile este şi ea numărabilă.

Această mulţime aparţine mulţimii tuturor funcţiilor parţiale $f : N \rightarrow N$, care este nenumărabilă (fiind practic izomorfă cu mulţimea numerelor reale). Rezultă că funcţiile calculabile sunt puţine (cardinal alef 0) din punctul de vedere al teoriei infiniţilor actuali iniţiată de G. Cantor.

În acest moment merită amintită cunoscuta teză (practic o conjectură) a lui Church-Turing, care, în esenţă, afirmă că pentru orice algoritm (procedură care se termină într-un număr finit de paşi) există o maşină Turing echivalentă.

Noţiunea de algoritm în sens clasic impune ca acesta să poată fi implementat pe o maşină Turing. Cu alte cuvinte, dacă un calculator poate implementa un algoritm (o funcţie intuitiv calculabilă), acesta poate fi implementat şi de o anumită maşină Turing. Reciproca nu este însă adevarată. Un algoritm implementat de o maşină Turing nu este în mod sigur implementabil pe un calculator real, cel puţin datorită resurselor infinite de memorie ale maşinii abstracte.

Această teză a adâncit înţelegerea noţiunii de algoritm de-a lungul timpului. În prelungirea acestei teze, ca o opinie personală, toate programele actuale care rulează pe un calculator - unele categorisite ca fiind "ultrasofisticate, adaptive, inteligente" etc. - sunt, în fond, destul de primitive în esenţă, de vreme ce se mapează şi se execută pe arhitecturile simpliste ale PC-urilor actuale.

Aşadar, toate aceste aplicaţii complexe sunt reductibile la o succesiune de operaţii extrem de simple şi care rulează pe arhitecturi propuse acum mai bine de 70 de ani. Aceste arhitecturi încă procesează nativ exclusiv instrucţiuni maşină relativ rudimentare (codificate binar) şi date binare, fără să aibă cunoştinţă de semantica algoritmilor aplicaţiei ori de cea a constructelor limbajului de nivel înalt etc.

În general, semantica aplicaţiei se pierde în urma compilării. Complexi-

tatea aplicaţiilor actuale provine din acumulări de natură cantitativă, fiind reductibilă în principiu la succesiuni de operaţii binare triviale.

Evident că această complexitate este uriaşă, având în vedere ierahizarea ei pe niveluri succesive, ele însele complexe şi care intercomunică (maşină, arhitectură, compilator, sistem de operare, limbaj de nivel înalt, aplicaţie inteligentă etc.). Orice aplicaţie, oricât de sofisticată, este mapabilă şi executabilă pe modelul de maşină Turing ori pe cel de maşină von Neumann, ambele de mare simplitate structural-funcţională.

Din păcate, actualmente paradigma aceasta nu mai este eficientă. Calitativ deci, progresele nu sunt totuşi chiar aşa de spectaculoase în ştiinţa şi ingineria calculatoarelor văzute ca ansamblu hardware-software. Ar fi deci necesară lărgirea paradigmei actuale a calculatoarelor de tip von Neumann la maşini inteligente nativ, intrinsec, prin structurile şi funcţiile lor de la nivelul cel mai de jos. Aceste maşini nu ar trebui doar să calculeze (cum fac astăzi!) ci şi, spre exemplu, să infereze logic, să aibă ontologii proprii, să recunoască forme, să raţioneze nuanţat etc. Nu insistăm aici asupra acestei idei.

Revenind la definiţia unui şir binar aleator, ar trebui ca un asemenea şir să nu fie generat de un algoritm. Cu alte cuvinte, secvenţa de biţi nu trebuie să fie generabilă printr-o funcţie Turing-calculabilă. Acest fapt ar fi echivalent cu a cere ca secvenţa să fie generată printr-o funcţie care nu este Turing-calculabilă.

Asociind secvenţele aleatoare cu funcţiile parţiale care nu sunt Turing-calculabile (şi care le-ar putea genera), rezultă că aceste secvenţe aleatoare sunt nenumărabile (de puterea continuului), deci majoritare pe mulţimea tuturor funcţiilor parţiale $f : N \rightarrow N$. Aparent paradoxal şi oarecum ironic, deşi sunt infinit-majoritare, nu se poate specifica niciuna!

Din păcate, această definiţie a secvenţelor aleatoare, bazată pe conceptul de calculabilitate, nu are o utilitate practică clară, de natură inginerească pentru că defineşte secvenţele aleatoare fără să le genereze efectiv. Pe de altă parte, generarea acestora ar fi în contradicţie cu aleatorismul lor.

În plus, se arată că există secvenţe necalculabile de simboluri dar care au regularităţi ce contrazic intuiţia noastră despre aleator [10]. S-ar putea face o analogie între noţiunile de calculabil-necalculabil şi respectiv cele de număr raţional-număr iraţional.

Este ştiut faptul că mulţimea numerelor raţionale este numărabilă, în timp ce mulţimea numerelor iraţionale este nenumărabilă. Mare parte din aceste numere iraţionale sunt aleatoare în conformitate cu definiţia lui Borel dar şi cu criteriul necalculabilităţii secvenţelor aleatoare, anterior prezentat.

Acest fapt, din păcate, nu are nicio utilitate practică pentru că aceste numere, cu foarte puţine excepţii ($\sqrt{2}, e, \pi, \log 2$ etc.), nu se pot exprima! Mai mult, în acest moment nici măcar nu se ştie dacă aceste constante matematice sunt Borel-normale sau nu (mai precis, este vorba de şirul zecimalelor acestor numere iraţionale sau, unele, chiar transcendente).

Oricum, aceste excepţii nu pot fi considerate aleatoare de vreme ce există algoritmi generici care le produc. Aşadar, aleatorul există, este majoritar, dar nu-l putem exprima!

4. Câteva concluzii şi idei de dezvoltare

Nu există încă o paradigmă universală satisfăcătoare pentru şirurile aleatoare de simboluri, problema fiind de actualitate şi de interes pentru multe categorii de specialişti, nu doar pentru matematicieni. Definirea şi înţelegerea aleatorului sunt, deloc surprinzător, legate strâns de noţiuni precum cele de calculabilitate, entropie informaţională, algoritmi, teoria complexităţii, teoria infinităţilor actuali etc.

Prezenţa ubicuă a aleatorului în ştiinţă şi în tehnologie, dar chiar şi în viaţa noastră de zi cu zi, trebuie acceptată ca atare, înţeleasă cât mai profund şi gestionată adecvat, cu implicaţii negative minimale. Altfel, aleatorul ne poate manipula în mod dăunător şi, uneori, chiar frustrant. Relativ la problema salturilor condiţionate impredictibile, se conturează următoarele idei practice de caracterizare a acestora din punct de vedere al gradului lor

de aleatorism:

- Acurateţea predicţiei unui şir de simboluri printr-un predictor HMM ar putea defini un anumit grad de aleatorism al secvenţei, din punct de vedere practic.

Desigur, se va modifica numărul de stări ascunse ale predictorului HMM astfel încât să se determine acurateţea maximă de predicţie. Este interesant dacă aceste predictoare puternice - nefezabile din punct de vedere al implementării în hardware - ar putea predicţiona cu succes salturile condiţionate de tip unbiased.

În caz afirmativ, ar rezulta că informaţia relevantă există dar este greu de identificat, diferind de la un salt la altul.

În caz negativ, gradul de aleatorism intrinsec al acestor branch-uri ar fi unul semnificativ.

- Credem că o măsură inginerească acceptabilă de definire a gradului de aleatorism aferent unei secvenţe binare S, bazată pe entropia discretă E(S), ar putea fi dată de produsul D(S)E(S).

- Rata de compresie a unei secvenţe de simboluri, obţinută prin algoritmi cunoscuţi de compresii fără pierderi (ex. gzip), ar putea constitui o altă măsură a aleatorismului secvenţei.

În cazul secvenţelor binare generate de salturile dificil predictibile, rata de compresie a acestora ar trebui să fie mai mică decât în cazul celorlalte branch-uri.

- Complexitatea Kolmogorov a secvenţei de program maşină care generează salturile condiţionate impredictibile ar putea constitui o altă metrică care să le caracterizeze aleatorismul. Astfel, complexitatea acestora ar trebui să fie mai mare decât a celorlalte salturi condiţionate. Cum complexitatea programelor creşte, este de aşteptat ca numărul şi influenţa salturilor unbiased să crească.

Sperăm că toate aceste idei, rezultate în urma studiului şirurilor şi secvenţelor aleatoare de simboluri, se vor concretiza şi deci, vor putea fi prezentate într-o viitoare lucrare.

Bibliografie

[1] *The 1st JILP Championship Branch Prediction Competition* (CBP-1), http://www.jilp.org/cbp, 2004

[2] Gellert A., Vinţan L., *Person Movement Prediction Using Hidden Markov Models, Studies in Informatics and Control*, Vol.15, No. 1, ISSN: 1220-1766, National Institute for Research and Development in Informatics, Bucharest, March 2006

[3] Gellert A., Florea A., Vinţan M., Egan C., Vinţan L. *Unbiased Branches: An Open Problem, Lecture Notes in Computer Science*, Advances in Computer Systems Architecture, vol. 4697, pg. 16-27, Springer-Verlag Berlin / Heidelberg, 2007

[4] Marcus S., *Paradigme universale*, Ed. Paralela 45, Piteşti, 2005, paginile 248-257

[5] Marcus S. - Comunicare personală (trimisă prin e-mail autorului), 23 ianuarie, 2008

[6] Rabiner L. R.,*A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, Proceedings of the IEEE, Vol. 77, No. 2, February 1989

[7] Vinţan L.,*Towards a High Performance Neural Branch Predictor*, Proceedings of the International Joint Conference on Neural Networks, Washington DC, USA, 10-16 July, 1999

[8] Vinţan L., Gellert A., Florea A., Oancea M., Egan C., *Understanding Prediction Limits through Unbiased Branches, Lecture Notes in Computer Science*, Advances in Computer Systems Architecture, vol. 4186, pg. 480-487, Springer-Verlag Berlin / Heidelberg, 2006

[9] Vinţan L., *Prediction Techniques in Advanced Computing Architectures*, Matrix Rom Publishing House, Bucharest, 2007

[10] Volchan S. B., *What Is a Random Sequence?*, The American Mathematical Monthly, 109, January 2002

[11] http://en.wikipedia.org/wiki/Kolmogorov_complexity

Universitatea "Lucian Blaga" din Sibiu
Catedra de Calculatoare şi Automatizări,
Str.E. Cioran, Nr. 4, Sibiu - 550025, ROMÂNIA,
E-mail: lucian.vintan@ulbsibiu.ro,
http://webspace.ulbsibiu.ro/lucian.vintan