Cuvânt înainte

Ediția din mai 1992 a cunoscutei publicații *ACM SIGPLAN Notices* a fost un moment de răscruce în istoria comunității programatorilor interesați de Haskell. Întregul număr din mai 1992 era dedicat numai Haskell-ului, astfel limbajul fiind prezentat tuturor într-un mod precis, profesionist și totodată unei largi audiențe. Erau publicate atunci și acolo două părți: (1) *Raportul Haskell* ; descrierea de referință a acestui nou limbaj funcțional, pur și non-strict, produs al unei echipe internaționale de experți în limbaje de programare (Comitetul Haskell) și (2) această *Mică introducere în Haskell 98* la scrierea căreia am lucrat împreună cu Joseph Fasel de la Laboratorul Național din Los Alamos.

Privind retrospectiv, constat că materialele care formează această pereche s-au sprijinit unul pe altul foarte bine fiind totodată de folos întregii comunități, primul oferind tot setul de detalii privind sintaxa și semantica limbajului, celălalt fiind un tutorial dedicat celor care sunt familiarizați cu limbajele de programare în general și cu cele funcționale în special. Unele voci au afirmat că este un manual oarecum abrupt, dar pe de altă parte el este relativ scurt, conține un set de exemple bune și s-a dovedit eficient în predarea noțiunilor fundamentale.

Într-adevăr, peste ani acest volum a avut drumul său propriu în viața comunității. John Petterson (pe atunci la Yale) a devenit al treilea autor, contribuind substanțial la completarea sa. A fost convertit în HTML și a devenit o componentă cheie a site-ului haskell.org. Pe măsură ce limbajul Haskell a evoluat, a evoluat și acest manual, prin munca diferiților colaboratori (de exemplu Reuben Thomas l-a revizuit ulterior, în iunie 2000.)

Iar acum ajunge să fie tradus și în limba română! Îmi face o mare bucurie să ofer această carte comunității programatorilor din România, care până acum o aveau la dispoziție doar în engleză și în franceză. Doresc să-i mulțumesc lui Dan Popa de la Universitatea “Vasile Alecsandri” din Bacău, care a făcut eforturi ca acest lucru să fie posibil.

Haskell-ul a fost este și rămâne încă un limbaj cu un design curajos, chiar judecându-l după exigențele zilelor noastre. Comitetul Haskell a proiectat limbajul optând pentru a-l înzestra cu puritate, evaluare amânată optimizată, funcții de nivel înalt și tipizare statică. Ca urmare, comportamentul programelor se poate studia și demonstra cu mijloace matematice. Sintaxa, îndelung dezbătută, face și ea parte din considerabilul effort depus pentru a face din Haskell un limbaj de-a dreptul frumos.

Din fericire, aceste decizii curajoase n-au dat naștere la stupefacție și blocaje ci la inovații. Unul dintre cele mai uluitoare lucruri despre Haskell este felul cum

acesta a evoluat. S-au adăugat extensii noi însă toate perfect încadrate în design-ul originar și perfect justificabile în baza unor principii matematice. Limbajul s-a maturizat repede – n-a mai fost Comnitetul Haskell cel care a făcut adăugirile - și se dezvoltă în continuare.

Rapiditatea schimbărilor ni se pare acum, privind retrospectiv, o adevărată goană. A existat însă un element ordonator al acestui univers: GHC-ul, compilatorul limbajului Haskell dezvoltat la Universitatea din Glasgow de Simon Peyton Jones, cel care, împreună cu Simon Marlow își continuă activitatea în cadrul departamentului Microsoft Research, de la Cambridge. Disponibilitatea lui Simon de a experimenta cu ideile venite din comunitate implementându-le rapid în GHC a făcut din acesta o platformă de test de un real folos pentru utilizatorii care aveau noi idei de îmbunătățire a limbajului, ajutând repede la finisarea acestor idei.

Printre conceptele inovatoare aduse de Haskell, două atrag îndeosebi atenția: *clasele de tipuri* și *monadele*. Ele sunt implementarea a două noțiuni fundamentale: *tipurile*, respectiv *descrierile de calculele abstracte*. Sistemul de tipuri din Haskell-ul de astăzi este capabil să exprime multe combinații de tipuri pe care nici nu ni le imaginam atunci când limbajul a fost proiectat: clase și familii de tipuri, dependențe funcționale, tipuri asociate, tipuri de date abstracte generalizate, tipuri cuantificate universal, recursii polimorfice, (…), tipuri abstracte, cuantificări de supratipuri, variabile de tip, clase de tip multiparametru și multe altele. Sistemul de tipuri din Haskell include un set uimitor de bogat de idei, în egală măsură intimidante și cuprinzătoare. Pe linia acestei dezvoltări a sistemului de tipuri, Sfântul Gral ar fi descoperirea unui design minimal, un cuplu de idei simple dar suficient de puternice ca să le subsumeze pe toate cele de mai sus.

Folosirea monadelor în Haskell, în special folosirea lor la operațiile de I/O a impus crearea unei sintaxe speciale, numită “do-notație” pentru a ușura programarea monadică. O monadă este un exemplu de structură de *calcule abstracte*, un șablon reutilizabil care surprinde esența desfășurării unor calcule. De exemplu functorii, (...), catamorfismele, monadele și acele funcții numite “arrows” - săgeți formează un set din ce în ce mai mare de calcule abstracte pe care un programator calificat le are la dispoziție pentru a-și scrie diversele programe.

Dintre acestea, două au fost considerate atât de importante încât s-au proiectat două seturi de reguli de sintaxă, special pentru a ușura folosirea lor. Una este “do-notația” folosită la programarea cu monade alta este “sintaxa săgeților” recomandată pentru programele care folosesc Arrows. Este prematur de spus

unde va duce acest șir de dezvoltări, dar cu siguranță apariția inovațiilor va continua.

Alan Perlis, primul câștigător al premiului Turing a scris în faimoasele sale “Epigrams of Programming”,

> "*Un limbaj care nu-ți schimbă felul de a gândi despre programare nu ți-e de folos să-l înveți.*”

Îmi exprim speranța că citirea acestei mici introduceri în Haskell vă poate face să meditați asupra felului în care programați, deschizând astfel niște uși despre care nici nu știați că există și conducându-vă la a scrie programe elegante, singurele care merită într-adevăr scrise.

Paul Hudak
Yale University
New Haven, CT USA
June 2011

Cuvânt înainte la ediția română

Această carte este în primul rând o restituire. Deoarece este vorba despre un volum ignorat, pentru o vreme, în România. Limbajul Haskell a apărut în anii 1987-88, într-o primă formă, iar acest volum a fost publicat în limba engleză la începutul deceniului 1990-2000. Era o perioadă de efervescență profesională în domeniul programării funcționale, perioada când la Universitatea din Edinburgh, profesorul Eugenio Moggi prezenta cursul „An abstract view of Programming Languages", curs în care atrăgea atenția asupra șablonului monadic de programare. În acea perioadă studiam Informatica la Univ. „Al.I.Cuza" din Iași, având ca profesor de programare funcțională pe lectorul, la acea dată, Gabriel Ciobanu, care ascultase cursurile profesorului Moggi, persoană de la care am învățat ML-ul și am prins gustul semanticilor și al limbajelor funcționale (fundamentate pe lambda calcul), acele limbaje în care totul este până la urmă exprimat în funcții, ceea ce înseamnă că, asemena unei funcții, programul va da întotdeauna același rezultat corect, pentru aceleași date de intrare.

După 2001, în cursul studiilor de doctorat, aflat în căutarea tehnicilor de implementare a limbajelor modulare cu componente interschimbabile – fie ele combinatori de parsere, arbori modulari, semantici denotaționale monadice înzestrate cu separabilitate am luat cunoștință de existența acestui limbaj funcțional, Haskell-ul, practic necunoscut în România și am fost impresionat de eficiența și eleganța acestuia. Era deja standardizat, ajunsese la standardul Haskell 98, iar această carte avea deja o ediție revizuită, la circa 7 ani de la apariție, ediția 1999 pe care am folosit-o ca bază pentru versiunea română. Se lucra pe atunci la ediția 2000 a acestei cărți.

Așa cum sublinia în introducerea sa prof. Paul Hudak, această carte, numită „Gentle Introduction to Haskell" nu era totuși, iar multe voci au confirmat, chiar atât de „gentle", fiind în multe privințe o carte cu mult deasupra nivelului începătorilor. În lumea software-ului liber, iar limbajul Haskell este un software liber din domeniul public, există obiceiul de a clasifica software-ul drept beta, chiar când funcționează mai stabil decât produsele comerciale, iar cărțile profesionale sunt intenționat etichetate drept „mică introducere". „Gentle Introduction to Haskell 98" n-a făcut excepție de la aceste reguli, fiind un manual care acoperă de la bazele limbajului și structurile de date fundamentale până la construcția serverelor și limbajelor incluse în Haskell. Sunt prezente de asemenea calculul matricial, clasele, polimorfismul, citirea și scrierea datelor compuse și exemple de programare monadică cu acțiuni în do-notația care amintește de C dar permite o flexibilitate dincolo de orice v-ați imaginat vreodată.

Pe de altă parte, anul universitar 20010-2011 s-a suprapus parțial cu anul european al voluntariatului și o serie de activități au purtat această marcă. Cum promisesem comunității Haskell din România o ediție a volumului „Gentle" am propus voluntarilor să se ocupe de traducerea a doar două pagini de persoană și am distribuit circa 50 de pagini de carte la circa 25 de voluntari. Restul volumului

urma să fie tradus și îngrijit de coordonator. Dintre aceștia cam jumătate au prezentat traduceri acceptabile, restul oferind doar pagini produse de Google Translator, neverificate, sau alte pagini de o calitate inutilizabilă. Rezultatele muncii voluntarilor implicați, parțial revizuite, formează capitolul semnal, prezentat on-line pe pagina de internet dedicată volumului (http://www.haskell.org/haskellwiki/Gentle). Adresez pe această cale mulțumiri întregii echipe de voluntari dintre care mă bucur să-i numesc pe: Silviu Pană, Sameș Loredana, Berlodean Elena, Marius Ailincăi, Halânga Victoria, Budău Mihai, Motan Cătălin, Rău Cătălina, Alexandru Maxinaș, Pavelescu Alexandru, Slabu Lavinia. Din echipa de voluntari au făcut parte, de asemenea Damian Constantin Sebastian, Obreja Sergiu, Vișniuc George Viorel, Bogdan Sucheană, Mihai Chiriac. La descreținea frunților echipei a contribuit accidental studentul P. V. a cărui contribuție realizată cu un translator automat a fost atât de amuzantă încât a meritat o pagină web specială. Dincolo de umorul involuntar pagina arată clar că „Gentle Introduction to Haskell" nu era o carte care să fie tradusă de azi pe mâine, ceea ce am constatat și atunci când, citind câte-o pagină greu traductibilă, am fost în situația s-o înlocuim complet cu explicații proprii, scrise de la zero.

Mulțumesc pe această cale autorilor ediției engleze, Paul Hudak John Peterson, Joseph H. Fasel pentru condițiile de licențiere ale volumului, acesta fiind oferit sub o licență care ne-a permis inclusiv modificarea integrală sau rescrierea lui, ceea ce, de altfel am și făcut unde am considerat că a fost nevoie.

Mulțumesc echipei editoriale de la editura Matrix Rom din București care s-a angajat să implice propriul lor capital și propriile lor talente în producerea și distribuirea acestor cărți despre limbajul Haskell. Păstrez și acum în memoria computerului și în amintire frumoasele și coloratele variante de coperți propuse rapid și profesionist de graficianul echipei, dintre care una ați văzut-o deja lunând în mâini cartea.

Urmare a amabilei aprobări, primită prin e-mail de la Prof. Simon Peyton Jones am inclus în ultimul minut un rezumat al ideilor importante despre *acțiunile de I/O* din Haskell, așa cum le-a prezentat dumnealui la o ediție a școlii de vară de la Marktoberdorf. Îi mulțumim totodată pentru tot ceea ce face și a făcut întru susținerea activității membrilor comunității Haskell.

Dan Popa
Universitatea „Vasile Alecsandri" din Bacău, România
Iunie 2011