1.2 Rezumat

Rezultatele obtinute in timpul doctoratului

Am obtinut doctoratul la Universitatea Nationala din Singapore in 2008, sub indrumarea Profesorului Wei-Ngan Chin. In activitatea mea de doctorat m-am concentrat pe dezvoltarea de analize avansate de program bazate pe tipuri pentru limbajele orientate pe obiecte [Cra08]. Lucrarea mea de doctorat propune urmatoarea teza: *o analiza simpla a fluxului adnotarilor care sunt partial ordonate si care decoreaza tipurile limbajelor de programare orientate pe obiecte poate genera sisteme de tipuri avansate cu beneficii practice importante*. Obiectivul principal al lucrarii mele de doctorat a fost demonstrarea acestei teze prin dezvoltarea unor sisteme de tipuri avansate bazate pe flux care imbunatatesc calitatea sistemelor software. In contextul unui limbaj orientat pe obiecte, de tip Java, am dezvoltat doua sisteme avansate de tipuri care imbunatatesc doua aspecte ale calitatii software-ului, si anume gestionarea corecta a memoriei prin intermediul tipurilor adnotate cu regiuni si reutilizarea codului prin intermediul tipurilor generice.

Primul sistem de tipuri [CCQR04, CQC08] pe care l-am dezvoltat asigura la compilare gestionarea corecta a memoriei heap organizata sub forma de regiuni. Am proiectat si dezvoltat *primul sistem de inferenta automata* a tipurilor adnotate cu regiuni in contextul paradigmei de programare orientata pe obiecte. Rezultatele experimentale au aratat ca programele care utilizeaza regiunile inferate de catre sistemul nostru sunt capabile sa reutilizeze memoria heap intr-o maniera comparabila cu ceea ce face un garbage collector in timpul executiei.

Cel de-al doilea sistem de tipuri [CGPC06] pe care l-am dezvoltat este o combinatie inovativa intre o analiza de flux si o modalitate de verificare modulara a tipurilor. Noul nostru sistem de tipuri este un model simplu dar precis pentru tipurile parametrice variante. Tipurile parametrice variante combina polimorfizmul bazat pe relatia de subtipare cu polimorfizmul parametric pentru a avea relatie de subtipare mult mai flexibila in cadrul paradigmei obiectuale. O caracteristica de baza a acestor tipuri este varianta. Experimentele noastre au demonstrat ca sistemul nostru de tipuri poate elimina mai multe downcast-uri decat sistemul de tipuri al limbajului Java.

Rezultatele obtinute ca si Postdoc la Universitatea Durham

Dupa terminarea doctoratului, am obtinut o bursa postdoctorala la Univeristatea Durham, in cadrul grupului Profesorului Shengchao Qin. In timpul acestei perioade, pe langa faptul ca am continuat sa lucrez in domeniul analizelor de program bazate pe tipuri, am devenit interesat de verificarea si analiza automata a modului in care programele utilizeaza memoria heap. Am realizat ca metodelor bazate pe tipuri sunt limitate si de aceea am inceput sa studiez metodele bazate pe logica, in special logica separation logic. In timpul acestei perioade am supervizat in co-tutela 2 studenti doctoranzi care dezvoltau aplicatii ale logicii separation logic: Guanhua He and Chenguang Luo. De asemenea, am finalizat implementarea proiectului Microsoft industrial (impreuna cu prof. Chin Wei-Ngan de la

NUS) privind o gestionare scalabila a memoriei impartia in regiuni pentru platforma .NET.

Primul rezultat a fost aplicarea metodei de inferenta a regiunilor, pe care am dezvoltat-o in teza mea de doctorat la programe CLI (.NET). Am extins SSCLI 2.0 cu un modul care gestioneaza memoria heap utilizand regiuni si am adaugat in CIL noi instructiuni pentru operatiile pe regiuni. Am implementat un sistem de inferenta a regiunilor care transforma automat programele CLI ce utilizeaza un garbage colector in programe care urilizeaza regiuni. Rezultatele au fost publicate la conferinta APLAS 2008 [SCC08]. Am masurat performanta programelor cu regiuni obtinute prin aplicarea metodei noastre de inferenta. Aceste programe au o viteza de executie semnificativ mai mare in special cand utilizeaza structuri de date complexe. Spre deosebire de garbace colector, timpii de manevrare a regiunilor sunt predictibili, de aceea ele se preteaza mai bine aplicatiilor in timp real.

Cel de-al doilea rezultat a fost un algoritm inovativ de inferenta pentru sistemul de tipuri parametrice variante din teza mea de doctorat. O caracteristica fundamentala a tipurilor parametrice variante este varianta stabilita in momentul utilizarii. In functie de modul in care cimpurile unei clase sunt utilizate, varianta poate determina o relatie de subtipare covarianta, contravarianta, invarianta sau bivarianta. Programatorii pot decide modul in care sunt utilizate campurile unei clase parametrice prin intermediul variantei cu care adnoteaza tipurile argument ale unei clase parametrice. Desi bibliotecile Java au fost modificate astfel incat sa utilizeze tipuri parametrice variante, totusi programatorii evita pe cat posibil utilizarea tipurilor parametrice variante datorita dificultatilor pe care le intampina la alegerea variantei. In trecut au fost propusi mai multi algoritmi de refactorizare automata a codului Java nongeneric in cod Java cu tipuri generice. Toti acesti algoritmi nu iau in considerare varianta si beneficiile utilizarii ei. Algoritmul propus de noi rezolva aceste limitari si infera automat atat tipurile generice argument cat si varianta asociata lor. Folosim o metoda inovativa bazata pe intervale, astfel incat fiecare tip parametric variant este transformat intr-un tip interval cu doua limite de tip, o limita inferioara pentru scriere si o limita superioara pentru citire. Algoritmul nostru este modular: analizeaza independent fiecare metoda generand un sumar de constrangeri. Am construit un prototip si am comparat rezultatele obtinute de catre sistemul nostru de inferenta automata cu tipurile parametrice variante pe care le-ar adnota manual un specialist in acest domeniu. In toate cazurile sistemul nostru a fost capabil sa infere cele mai bune tipuri generice. Rezultatele au fost publicate la conferinta ESOP 2009 [CCHQ09].

Cel de-al treilea rezultat reprezinta o solutie inovativa pentru o problema practica: verificarea programelor care invoca proceduri necunoscute, adica proceduri ale caror cod nu este disponibil. In majoritatea cazurilor cand se verifica un program, nu intregul cod al acelui program este disponibil. De aceea acele parti de cod care sunt necunoscute reprezinta o mare provocare pentru procesul de verificare. Rezultatele noastre au fost publicate in workshop-ul ETAPS 2009 WING [CLH+09], la conferinta ICFEM 2010 [QLH+10b] si in jurnalul Journal of Symbolic Computation 2010 [LCQ+10]. Pentru solutia noastra am considerat situatia verificarii complete a corectitudinii functionale a unor programe ce utilizeaza structuri de date bazate pe pointeri. Plecam de la o specificatie in maniera Hoare $\{\Phi_{pr}\}$ prog $\{\Phi_{po}\}$ unde programul prog contine apeluri la o procedura necunoscuta unknown si inferam specificatia $mspec_u$ pentru unknown din contextele de apel. Astfel problema verificarii

programului prog poate fi redusa la problema demonstrarii ca procedura unknown (odata ce codul sau devine disponibil) satisface specificatia inferata $mspec_u$. Specificatia $mspec_u$ care va trebui sa fie respectata de catre procedura unknown se calculeaza automat utilizand o analiza abductiva a formei structurilor de date. Analiza este proiectata pentru un domeniu abstract combinat. Cu ajutorul prototipului nostru am facut mai multe experimente care au validat viabilitatea solutiei propuse de noi.

Rezultatele obtinute ca si Research Fellow la Universitatea Nationala din Singapore

Dupa terminarea perioadei de postdoctorat, am obtinut o pozitie de research fellow la Universitatea Nationala din Singapore. In timpul acestei perioade, am continuat activitatea mea de cercetare in domeniul logicii separation logic. Impreuna cu colegii mei am dezvoltat un calcul inovativ de specializare pentru procesele de verificare si analiza. Rezultatul a fost publicat la conferinta CAV 2011 [CGV+11]. Mecanismele de abstractizare bazate pe logica separation logic combinate cu predicate inductive definite de utilizator reprezinta un mijloc puternic si expresiv pentru specificarea structurilor de date din heap cu proprietati invariante. Totusi expresivitatea are un cost:in general satisifiabilitatea si verificarea implicatiilor acestor tipuri de logici necesita expandarea unor predicate disjunctive ceea ce implica un proces costisitor de cautare a demonstratiilor. Pentru a adresa aceasta problema am propus o tehnica de *specializare a predicatelor* care permite o eliminare simbolica a acelor disjuncti care sunt infezabili in interiorul instantei unui predicat fara a fi necesara deschiderea corpului predicatului prin unfoldig sau inlining. Tehnica noastra este prezentata ca si un calcul a carui rezultate mentin satisfiabilitatea formulelor si reduc costul de manipulare a formulelor. Rezultatele noastre experimentale au confirmat reducerea semnificativa a timpului de manipulare a predicatelor in cazul folosirii tehnicii noastre de specializare. Specializarea este o tehnica comuna pentru optimizarea codului, totusi utilizarea ei in procesele de verificare si analiza este o noutate. Tehnica propusa de noi nu este limitata doar la abastractiile bazate pe logica separation logic, ea poate fi aplicata pe orice domeniu abstract care are predicate disjunctive.

Rezultatele obtinute ca si cercetator vizitator la Universitatea Teesside

Dupa angajarea la Universitatea Babes-Bolyai, am fost pentru o scurta perioada de timp cercetator vizitator la Universitatea Teesside din Anglia. In timpul acestei vizite, impreuna cu colegii de acolo am stabilit o noua directie de cercetare: descoperirea automata a specificatiilor exprimate intr-un domeniu combinat pentru programele care utilizeaza structuri de date din heap. Rezultatele au fost publicate la conferinta ICFEM 2013 [HQCC] si apoi in jurnalul Science of Computer Programming 2017 [QHC+17]. Descoperirea automata a specificatiilor exprimate intr-un domeniu combinat pentru programele care utilizeaza structuri de date din heap este un task complex datorita aliasingului si mutabilitatii structurilor de date. Taskul se complica si mai mult datorita domeniului combinat care contine informatii despre forma structurilor de date, informatii numerice si informatii din teoria multimilor despre continutul structurilor. Am propus o analiza compozitionala care calculeaza

un sumar pentru fiecare metoda, sumar care este independent de cum acea metoda este apelata. Analiza noastra se bazeaza pe o metoda noua de abstractizare care foloseste bi-abductia in doemniul combinat si in prezenta predicatelor inductive definite de utilizator. Analiza a reusit sa descopere automat pre-/post-conditii care nu au putut fi inferate automat inainte. Pe linaga faptul ca analiza noastra demonstreaza proprietati de utilizare sigura a memoriei, ea deasemenea descopera relatii intre domeniile pure si cele ce se refera la forma structurilor astfel facand un pas important spre demonstrarea completa a corectitidinii functionale a programelor. Experimentele pe care le-am facut cu prototitpul implementat de noi au aratat ca metoda propusa de noi poate descoperi proprietati foarte interesante chiar si in cazul programelor complexe.

Rezultatele obtinute la Universitatea Babes-Bolyai

M-am angajat la Universitatea Babes-Bolyai in anul 2013 ca si lector. Apoi am fost promovat ca si conferentiar. In aceasta perioada, am continuat activitatea mea de cercetare in domeniul sistemelor de tipuri precum si in domeniul logicii separation logic. Am stabilit si o noua directie de cercetare: logica sesiunii. Am co-supervizat trei studenti doctoranzi: Tibor Kiss (de la inceput pana la sustinerea tezei) si Gabriel Glodean la Universitatea Babes-Bolyai si Andreea Costea la Universitatea Nationala din Singapore. De asemenea, am introdus mai multe cursuri nivel licenta si cursuri nivel master despre analiza si verificarea programelor.

Primul meu rezultat a fost elaborarea unui calcul al erorilor *must* (care sigur vor aparea) si *may* (care ar fi posibil sa apara). Acest calcul l-am folosit pentru dezvoltarea unui nou mecanism de specificare a proprietatilor de siguranta (corecte si/sau complete) de catre utilizatori. Rezultatele au fost publicate la conferinta NASA Formal Methods Symposium 2013 [LSCC13]. Calculul dezvoltat de noi constituie fundamentul pentru demonstrarea corectitudinii unui program si/sau pentru descoperirea de defecte intr-un program. La baza calcului nostru este o algebra cu o latice cu patru posibile situatii in care se pot afla posibilele stari ale unui program (si anume *nu se poate afla*, *valid*, *eroare ce apare sigur* si *eroare ce ar putea apare*). Algebra contine si *patru operatori* pentru a calcula situatia in care se afla programul. Am aratat cum calculul nostru poate suporta *demonstrarea corectitudinii* si *localizarea erorilor*. Calculul nostru poate fi extins la logica *separation logic* care suporta leme si predicate definite de utilizator. Am implementat si integrat calculul nostru in cadrul unui sistem de verificare automata a programelor ce utilizeaza pointeri la memorie. Experimentele, pe care le-am efectuat au confirmat ca realizarea celor doua obiective ale calcului nostru (adica demonstrarea corectitudinii si cautarea defectelor) se face cu un cost aditional nesemnificativ.

Cel de-al doilea rezultat a fost elaborarea unei logici cu disjunctii, denumita logica sesiunii pentru a specifica si verifica implementarea protocoalelor de comunicatie. Asigurarea corectitudinii si sigurantei programelor centrate pe comunicatie este un task foarte important dar si foarte complex. Dezvoltarea aplicatiilor software moderne necesita mecanisme expresive de specificare si verificare a comunicatiei dintre diferitele subcomponente. In ultimii zece ani, majoritatea propunerilor pentru a caracteriza diferitele aspecte ale counicatiilor structurate s-au bazat pe tipuri sesiune. Sistemele

de tipuri sesiune permit o specificare intr-o maniera concisa a protocoalelor de comunicatie si o modalitate simpla de verificare a implementarii lor. Noi am propus o solutie diferita de sitemele de tipuri, o solutie bazata pe logica. Logica propusa intial de noi era limitata la canale de comunicatie cu numai doi participanti. Totusi ea era capabila sa modeleze in mod natural mecanismul de delegare prin utilizarea canalelor de ordin superior. Logica noastra inglobeaza disjunctii si de aceea poate modela direct alegerile interne si externe utilizand doar simple instructiuni conditionale. In contrast, metodele anterioare aveau nevoie de cosntructii tip switch speciale. Propunerea noastra este o extensie a logicii separation logic si de aceea suporta programe ce utilizeaza memoria heap precum si programe fara stare ce transmit mesaje. Am demonstrat expresivitatea si aplicabilitatea logicii noastre pe mai multe exemple clasice din literatura. Rezultatele au fost publicate la conferinta ICECCS 2015 [CKC15].

Cel de-al treilea rezultat reprezinta o imbunatatire a versiunii initiale a logicii sesiunii. Am extins logica noastra pentru a suporta specificarea protocoalelor de comunicatie intre mai mult de doi participanti, precum si pentru verificarea conformitatii protocoalelor si a corectitudinii implementarii lor. Propunerea noastra combina idei folosite de sistemele de tipuri sesiune pentru participanti multipli pentru exprima concis protocoalele de comunicatie cu idei folosite de logica separation logic pentru a exprima accesul concurent la resursele comune (cum ar fi memoria heap). Logica noastra imbunatateste expresivitatea si precizia sistemelor de tipuri sesiune fara a sacrifica stilul lor concis. Pe de alta parte, metoda propusa de noi este mai precisa si se integreaza natural in procesul de verificare. Metoda noastra se paote aplica modular si automat. Rezultatul a fost publicat la conferinta APLAS 2018 [CCQC18].

Cel de-al patrulea rezultat a fost elaborarea unui sistem de tipuri pentru limbajele similare cu Java, sistem ce combina tipurile variante parametrice (adica tipurile generice) cu tipurile regiune. Noile tipuri le-am denumit tipuri regiune variante si reprezinta o generalizare a celor doua sisteme de tipuri propuse in teza mea de doctorat. Rezultatul a fost publicat la conferinta ICECCS 2018 [CCQ18]. Propunerea noastra se bazeaza pe o analiza a fluxului de executie. Analiza este ghidata de adnotarile de varianta. Sistemul de tipuri regiune variante garanteaza ca un program bine tipat utilizeaza o stiva de regiuni lexicale pentru face alocarea obiectelor si nu creeaza niciodata referinte care pointeaza spre nimic. Sistemul propus permite ca fluxul sa poate fi specificat si verificat modular la nivelul fiecarei metode. Sistemul de tipuri regiune elaborat in teza mea de doctorat era limitat la tipuri non-generice. De aceea acest nou sistem deschide oportunitati noi de a aplica rezultatele noastre anterioare la noile cadre de aplicatii Java destinate aplicatiilor care lucreaza cu un volum mare de date. Am inceput dezvoltarea unui prototip al sistemului nostru de tipuri regiune variante in cadrul compilatorului Java Jamaica. Primele experimente au confirmat ca timpul de verificare a utilizarii regiunilor este foarte mic. Verificarile tipurilor regiune care se fac in timpul compilarii permit eliminarea verificarilor din timpul executiei pe care masina virtuala Jamaica le foloseste sa garanteze ca regiunile de memorie sunt utilizate in siguranta.

Ultimul rezultat se refera la elaborarea unui cadru de generare automata a codului de gestionare a regiunilor de memorie care respecta specificatia Java pentru sisteme in timp real. In ultima perioada, au fost dezvoltate si utilizate in aplicatii critice mai multe masini virtuale Java care respecta specificatia

Java pentru sisteme in timp real. Gestionarea meomoriei bazata pe regiuni este o caracteristica de baza a specificatie Java pentru sisteme in timp real. Pornim de la un program Java adnotat cu tipuri regiune pe care aplicam trei analize bazate pe tipuri. Tipurile regiune sunt generate automat de catre sistemul nostru de inferenta sau sunt scrise de catre programator si verificate de catre sistemul nostru de verificare. Primele doua analize simplifica tipurile regiune, iar ultima analiza genereaza codul final repectand biblioteca Java pentru sisteme in timp real. Rezultatele preliminare au fost publicate la conferinta SYNASC 2018 [CG18].

Organizarea Tezei

Teza este organizata in functie de cele trei mari directii de cercetare ale mele, dupa cum urmeaza. Rezultatele obtinute in domeniul sistemelor de tipuri sunt descrise in Capitolul 2. Rezultatele obtinute in domeniul logicii separation logic sunt descrise in Capitolul 3. Rezultatele obtinute in domeniul logicii sesiune sunt descrise in Capitolul 4. In Capitolul 5, am prezentat planul nostru de cariera.