Dispozicija doktorske disertacije

Žiga Lukšič
Mentor: dr. Matija Pretnar

KAKO SOSEDU UKRASTI SENDVIČ BREZ DA OPAZI (HOW TO STEAL YOUR NEIGHBOURS SANDWICH WITHOUT BEING NOTICED)

Pregled znanstvenega področja in dosedanjih raziskav

Ena od prednosti funkcijskih programskih jezikov je močno matematično ogrodje. Programe lahko predstavimo kot matematične objekte in s tem pridobimo boljši vpogled v delovanje in močnejša orodja za analizo in dokazovanje lastnosti programov. Vendar ker matematične funkcije nimajo računskih učinkov, kot na primer pisanje teksta v datoteke ali komunikacija z oddaljenim strežnikom, računske učinke tudi težje smiselno umestimo v funkcijske programske jezike.

Eden od pristopov, poleg že uveljavljenega monadičnega pristopa k računskim učinkom [4], je uporaba algebrajskih učinkov in prestreznikov. Algebrajski učinki ne zahtevajo spremembe pisanja kode in jih preprosto komponiramo, kar je eden glavnih očitkov monadičnega pristopa.

Raziskave področja so se začele zgolj z uvedbo algebrajskih učinkov [5, 6], temu pa je kasneje sledil razvoj algebrajskih prestreznikov [7], ki posplošijo princip delovanja prestreznikov napak z možnostjo nadaljevanja izvajanja. Izkaže se, da lahko z algebrajskimi prestrezniki modeliramo večino želenih računskih učinkov, prav tako pa je ekvivalenten monadičnemu pristopu.

Uporaba algebrajskih učinkov prav tako omogoča sledenje računskim učinkom skozi sistem tipov. Pri tem želimo ohraniti lastnosti kot so polimorfizmi in varnost tipov. Razvoj sistema tipov z učinki je šel v smeri t.i. 'row-types' [3] in pa 'subtyping' [8].

Ključnega pomena za razumevanje in dokazovanje v programskih jezikih so pristopi, kjer programom podamo primeren matematičen pomen. Funkcijski programski jeziki uporabljajo strukturno denotacijsko semantiko, ki sloni na uporabi teorije domen in teorije kategorij.

tako se lepo sklopijo z obstoječo denotacijsko semantiko funkcijskih programskih jezikov [2], kar nam ponuja vpogled v matematični pomen programov.

Opis raziskovalne vsebine

Razvoj zanesljivih in ekspresivnih sistemov tipov skrbi tako za varnost programov kot tudi za dokazovanje lastnosti in pravilnosti programske kode. Z uporabo algebrajskih učinkov lahko sistem tipov razširimo na način, ki nam omogoča natančno sledenje računskih učinkov. Hkrati pa nam teorija algebrajskih učinkov nudi tudi enačbe, ki veljajo med učinki. Sistem tipov učinkov tako razširimo na sistem tipov teorij, ki poleg možnih učinkov hranijo tudi enačbe teorije. Te enačbe nam omogočajo lažje dokazovanje lastnosti programske kode, dodatno varnost s pomočjo iskanja napak v kodi, in odpirajo vrata novim pristopom k optimizaciji kode s pomočjo enačb.

Raziskovalna vprašanja

Želim pokazati, da implementacija enačb v programski jezik z algebrajskimi učinki ohrani vsaj del strukture teorije učinkov. Nadalje želim raziskati lastnosti, katerim mora zadoščati programski jezik, da v pripadajoči denotacijski semantiki ohranimo čim več koristnih lastnosti teorije. Poleg razvoja matematičnega ogrodja za teorijo učinkov in konstrukcije primernih logik za dokazovanje, nameravam sistem tudi uporabiti na primernih problemih, kot recimo analiza statističnih modelov. Pri modeliranju s pomočjo Bayesove statistike lahko zahtevane invariante izrazimo s pomočjo enačb, za katere upam, da jih lahko izrazimo v teoriji učinkov. Vse konstrukcije nameravam formalizirati z uporabo dokazovalnikov, ter s pomočjo tega dokazati pravilnost razširitve.

Pričakovani rezultati in prispevek k znanosti

Kljub temu, da se je teorija algebrajskih učinkov uveljavila v sferi funkcijskega programiranja, ostaja področje eksplicitnih enačb med učinki še dokaj neraziskano področje. Smer moje raziskave napram sorodnim [1] zahteva manj specializiran sistem tipov in je bolj v skladu z izvornimi idejami [7]. Dopolnitev tipov z enačbami med učinki olajša dokazovanje lastnosti programov, kar je ključnega pomena v mnogih uporabnih aspektih, kot npr. pri modeliranju s pomočjo Bayesove statistike, kjer je varnost naše kode ključnega pomena za kredibilnost rezultatov.

Literatura

[1] D. Ahman. Handling fibred algebraic effects. *PACMPL*, 2(POPL):7:1–7:29, 2018.

[2] A. Bauer and M. Pretnar. An effect system for algebraic effects and handlers. *Logical Methods in Computer Science*, 10(4), 2014.

[3] D. Hillerström and S. Lindley. Liberating effects with rows and handlers. In J. Chapman and W. Swierstra, editors, *Proceedings of the 1st International Workshop on Type-Driven Development, TyDe@ICFP 2016, Nara, Japan, September 18, 2016*, pages 15–27. ACM, 2016.

[4] E. Moggi. Notions of computation and monads. *Inf. Comput.*, 93(1):55–92, 1991.

[5] G. D. Plotkin and J. Power. Adequacy for algebraic effects. In F. Honsell and M. Miculan, editors, *Foundations of Software Science and Computation Structures, 4th International Conference, FOSSACS 2001 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2001 Genova, Italy, April 2-6, 2001, Proceedings*, volume 2030 of *Lecture Notes in Computer Science*, pages 1–24. Springer, 2001.

[6] G. D. Plotkin and J. Power. Algebraic operations and generic effects. *Applied Categorical Structures*, 11(1):69–94, 2003.

[7] G. D. Plotkin and M. Pretnar. Handlers of algebraic effects. In G. Castagna, editor, *Programming Languages and Systems, 18th European Symposium on Programming, ESOP 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*, volume 5502 of *Lecture Notes in Computer Science*, pages 80–94. Springer, 2009.

[8] A. H. Saleh, G. Karachalias, M. Pretnar, and T. Schrijvers. Explicit effect subtyping. In A. Ahmed, editor, *Programming Languages and Systems - 27th European Symposium on Programming, ESOP 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings*, volume 10801 of *Lecture Notes in Computer Science*, pages 327–354. Springer, 2018.