MUNI
FAKULTA
INFORMATIKY

POSUDEK OPONENTA
DIPLOMOVÉ PRÁCE

Název práce:	*Local Type Argument Synthesis for Erlang*
Autor práce:	*Bc. David Pavlík*
Vedoucí práce:	*doc. Mgr. Jan Obdržálek, PhD.*
Oponent:	*RNDr. Martin Jonáš, Ph.D.*

Cílem diplomové práce bylo rozšířit algoritmus *Local Type Argument Synthesis (*LTAS*)* tak, aby fungoval nad graduálním typovým systémem pro jazyk Erlang. Student také měl takto rozšířený algoritmus implementovat do nástroje pro typovou kontrolu Gradualizer, který podporuje programy v jazyce Erlang s částečnými typovými anotacemi.

S potěšením konstatuji, že tento cíl práce byl beze zbytku splněn. Student rozšířil algoritmus LTAS o často používaně typové konstruktory jazyka Erlang a o podporu dynamického typu „?“, díky čemuž je možné algoritmus použít v graduálním typovém systému. Student také implementoval algoritmus do nástroje Gradualizer a vyhodnocení ukazuje, že díky lokalitě algoritmu LTAS nástroj Gradualizer nyní produkuje přesnější a více informativní chybové hlášky.

Práce je psaná dobrou angličtinou s minimálním množstvím překlepů a gramatických chyb. Typograficky je práce na dobré úrovni, kterou místy kazí přetoky za tiskové zrcadlo strany. Struktura práce je logická a student v práci dobře vysvětluje většinu potřebných konceptů, aniž by při tom zabíhal do zbytečných detailů. V práci jsem našel několik menších chyb v matematickém textu (např. definice konzistentního podtypu nemá být $S\big|_T <: S\big|_T$, ale $S\big|_T <: T\big|_S$ nebo na straně 27 dole má místo $(\texttt{number}) \to \texttt{integer}$ být $(\texttt{number}) \to \texttt{number}$). Tyto chyby ale jsou lehce opravitelné a nijak zásadně nekomplikují porozumění textu.

Předložená práce převyšuje standard diplomových prací na Fakultě informatiky a zabývá se technicky náročným tématem, které navíc student samostatně rozvíjí nad rámec současného stavu poznání. V práci student píše, že v době odevzdání ještě nebyl odpovídající *pull request* na začlenění nové implementace do oficiální verze nástroje Gradualizer schválený. V současnosti už schválený je. Musím podotknout, že reakce správců nástroje v uvedené žádosti jsou *„I've just gone through the wiki description – it's excellent!“*, *„This [pull request] is a fantastic piece of work, it's truly impressive!“* a *„I don't have time to review properly, but I agree it's impressive.“* Jsem si také poměrně jistý, že pokud student dokončí všechny budoucí kroky, které v práci popisuje (zejména důkaz úplnosti implementovaného algoritmu), bude mít výsledná práce publikační potenciál.

Během obhajoby bych rád slyšel odpovědi na následující otázky:

- Na straně 17 píšete *„standard properties of subtyping relations, namely reflexivity [...] are not trivially evident“*. Proč není reflexivita triviálně zřejmá z pravidla S-REFL?
- Nemá na straně 37 v pravidle CG-ANYFUN v předpokladech místo $\overline{X} \vdash \overline{?} \lesssim \overline{S}$ být $\overline{X} \vdash \overline{S} \lesssim \overline{?}$ (a naopak v a CG-FUNANY), protože $\overline{S}$ se v závěru vyskytuje v kontravariantní pozici?
- Na straně 46 píšete, že pro rigidní typové proměnné jste neimplementoval žádná pravidla pro relaci být podtypem kromě reflexivity, největšího prvku, nejmenšího prvku a `any()`, protože *„users of the rigid variables cannot assume anything about them“*. Proč nemůže uživatel například předpokládat, že rigidní typová proměnná A splňuje $A <: (A \vee B)$?

- V textu zmiňujete, že současná implementace je jen parciálně korektní kvůli pravidlům CG-UNIONRIGHT*. Také zmiňujete, že implementace není korektní kvůli monomorfizaci argumentů polymorfních funkcí (poslední odstavec na straně 49). Jsou v implementaci ještě nějaké další důvody nekorektnosti nebo neúplnosti, o kterých víte?

Celkově navrhuji práci uznat jako diplomovou a z výše uvedených důvodů ji navrhuji hodnotit známkou výborně (A). Také navrhuji komisi zvážit ocenění práce *cenou děkana Fakulty informatiky Masarykovy univerzity*.

Brno 9. června 2024 Martin Jonáš