Posudek oponenta bakalářské práce

Student: Kraut Daniel
Téma: Automatická tvorba testovacích případů z datových toků (id 18777)
Oponent: Fiedor Tomáš, Ing., UITS FIT VUT

1. Náročnost zadání **obtížnější zadání**

Zadání jako takové je obtížnější. Vyžaduje pochopení pokročilých technik statické analýzy kódu a propojení několika netriviálních frameworků pro efektivní implementaci.

2. Splnění požadavků zadání **zadání splněno s vážnými výhradami**

Zadání bylo splněno s vážnějšími výhradami k bodům 1 a 4.
Nastudování a prezentace související tématiky zmiňuje pouze minimum souvisejících technik a nástrojů (blíže viz bod 4.). I když se jedná o BP, tak oblast statické analýzy kódu je velice dobře zmapovaná a zaslouží si podstatně více prostoru.Experimentální vyhodnocení v technické zprávě nedemonstruje dostatečně funkčnost generátoru na vybraném kritériu (blíže viz bod 4.).

3. Rozsah technické zprávy **je v obvyklém rozmezí**

Vysázená práce čítá 30 stran. Rozsah považuji v normě, blížící se však k minimu---řadu obrázků by bylo možné vysázet v kompaktnější formě (dohromady celkem 10 stran) nebo odlišnou reprezentací a některé navíc působí dojmem vaty. Kapitola 3.3. (popis jazyka C) není stěžejní pro prezentaci vytvářeného nástroje.

4. Prezentační úroveň předložené práce **30 b. (F)**

Prezentační úroveň práce obsahuje řadu nedostatků. Hlavním nedostatkem je slabý popis koncepce nástroje v kapitole 4. Kapitola sice zahrnuje poměrně rozsáhlý diagram 4.5., nicméně bez podrobnějšího popisu, co jednotlivé části reprezentují a jak program funguje jako celek. Kapitola pouze obsahuje stručné popisy modulů. Algoritmus na str. 19 mohl být vysázen jako algoritmus a ne jako seznam. Obrázky v sekci 4.1.1 jsou těžce pochopitelné a působí dojmem vyplnění místa.
Splnění bodu 1 pak vidím jako nedostatečné. Kapitola o související tématice a podobných nástrojích má pouze 4 strany a současně opomíjí řadu souvisejících technik jak pro automatizované testování tak získávání data-flow pomocí statické analýzy (např. náhodné testování, formy data-flow analýz, symbolická exekuce---na niž je mimochodem založen nástroj KLEE, který je zmíněn).

Další problém pak vidím v nepřesné (místy i chybné) prezentaci související problematiky:

1. Program KLEE "nezkouší co nejvíce vstupních hodnot a následně s nimi program spouští", ale je založen na symbolické exekuci vstupního programu (tzn. stále se jedná o statickou analýzu kódu). Výstupem pak jsou skutečně vstupy pro program, které se snaží o maximální pokrytí všech cest v programu (mimochodem na vysokém pokrytí si právě autoři zakládají).
2. Na rozdíl od zdroje [10] práce popisuje základní přístupy k testování mírně odlišně (nebo případně zmateně). Systém, ke kterému máme plný přístup, pořád můžeme testovat pomocí tzv. černé skříňky.
3. Práce je o **automatizované tvorbě testovacích případů**, takže uvádět jednotkové testy je skutečně bezpředmětné.

Experimentální vyhodnocení v kapitole 5 zahrnuje pouze 4 triviální příklady bez bližšího komentáře. Bez měření např. rychlosti implementace, pokrytí zdrojového kódu generovanými testovacími případy, či libovolnou jinou evaluací (např. objevení chyby, objevení netestované hraniční hodnoty). Z prezentovaných experimentů nejsem sto rozhodnout zda nástroj skutečně splňuje vybrané testovací kritérium (All-Defs Coverage). Vzhledem k tomu, že se jedná o téma z oblasti **Testování**, tak by tato část práce měla být skutečně důkladně zpracovaná.

5. Formální úprava technické zprávy **50 b. (E)**

Práce obsahuje řadu typografických chyb (např. jednopísmenná slova na konci řádků, chybné použití spojovníku, špatně sázené seznamy). Práce rovněž obsahuje několik překlepů (přepisy, chybějící slova) a gramatických chyb (v interpunkci).
Řada vět má špatný slovosled. Student navíc často používá nevhodná spojení a vatu.

6. Práce s literaturou **40 b. (F)**

Práce cituje 14 zdrojů, které jsou relevantní k řešenému problému. Jedná se o jednu monografii a jednu technickou zprávu, zbytek jsou internetové odkazy (např. server Wikipedia). Mnoho relevantních zdrojů však není v práci pokryto.

Více než vhodné by bylo ocitovat důležitý pojem CSP a Mini-Zinc Challenge (minimálně pod čarou).

7. Realizační výstup

50 b. (E)

Práce implementuje pouze jedno vybrané testovací kritérium (All-Defs Coverage). Prvotní kompilace práce podle přiloženého návodu nebyla úspěšná (problém s výjimkami v knihovně gecode), bylo nutno tento proces modifikovat. Navíc samotný proces kompilace je příliš zdlouhavý. Doporučovaný postup zahrnuje kompletní kompilaci LLVM i překladače Clang se všemi extra nástroji (např. clang-tidy, apod.), které nejsou potřebné. Výsledek je tak nejen zdlouhavý, ale i s vysokou spotřebou paměti (výsledek má přes 20GB).
Vytvořený nástroj korektně reprodukuje demonstrační příklady z technické zprávy. Nicméně existují triviální příklady, na kterých nástroj cyklí (např. s podmíněnými příkazy return ve funkci bez lokálních proměnných).

Z hlaviček přiložených souborů není jisté, zda je autor práce skutečně autorem (trochu nechápu, proč obsahují licenci Gecode + LLVM).

8. Využitelnost výsledků

Vzhledem ke stavu technické zprávy i implementaci nepovažuji aktuální využití práce v praxi.

9. Otázky k obhajobě

1. Bylo by možné (a pokud ano, tak jak) rozšířit Váš přístup pro jiné front-endy LLVM frameworku (např. pro Javu)?
2. Diskutujte možné využití Vašeho nástroje/přístupu v dalších oblastech analýzy programů, např. pro hledání chyb apod.

10. Souhrnné hodnocení

45 b. nevyhovující (F)

Vzhledem k tomu, že dle mého názoru, nebyl dostatečně splněn první a čtvrtý bod zadání, technická zpráva je nekvalitní a výsledný program má několik nedostatků, tak navrhuji výsledné hodnocení **nedostatečné (F)** .

Prohlášení: Uděluji VUT v Brně souhlas ke zveřejnění tohoto posudku v listinné i elektronické formě.

V Brně dne: 27. května 2016

.....
podpis