SOURCE-CODE MIGRATION USING DECOMPILATION

Tomáš Korec
Master Degree Programme (3), FIT BUT
E-mail: xkorec00@stud.fit.vutbr.cz

Supervised by: Petr Zemek
E-mail: izemek@fit.vutbr.cz

Abstract: This paper deals with source-code migration of high-level programming languages using decompilation. The developed migration tool is built on top of the middle-end and back-end of Lissom project decompiler. Several compilers generating LLVM IR code from an input language are discussed. Suitable compilers were integrated to the migration tool. Compiled LLVM IR code is an input of the decompiler's optimizing middle-end. Output from the migration tool is a code in the C language or Python-like language generated by the back-end of the decompiler. Input languages are Fortran and its dialects, C/C++/Objective-C/Objective-C++, and D. The paper describes ways to improve quality and readability of the produced source code.

Keywords: migration, source code, decompilation, Lissom, LLVM IR, Fortran, C/C++, D, Objective C/Objective C++

1 ÚVOD

V tejto práci, ktorá bola vypracovaná v rámci projektu Lissom, sa rozoberá problematika migrácie zdrojových kódov z jedného vysoko úrovňového programovacieho jazyka do druhého. Cieľom práce je vytvoriť migračný nástroj podporujúci viacero vstupných a viacero výstupných jazykov za použitia existujúceho dekompilátoru projektu Lissom.

V dnešnej dobe ešte stále existuje množstvo projektov, ktorých hlavné časti sú implementované v zastaralých jazykoch alebo ich dialektoch. Udržovanie týchto častí je náročné a nákladné, potrebné nástroje ako prekladač a ladiaci nástroj nemusia byť ďalej vyvíjané a podporované. Navyše odborníci na zastaralé jazyky postupom času ubúdajú. Migrácia zdrojových kódov je často jediným riešením [1].

2 MIGRÁCIA ZDROJOVÝCH KÓDOV

Ručná migrácia zdrojových kódov je pomerne náročný proces, ktorý je časovo veľmi náročný a generuje množstvo nových chýb. Preto vznikajú snahy vytvoriť poloautomatické alebo automatické nástroje na uľahčenie tohto procesu. Dosiaľ vyvinuté nástroje často nie sú úplne automatické, ale ich hlavnou nevýhodou je obmedzená podpora vstupných a výstupných jazykov. Väčšinou ide o nástroje podporujúce jeden vstupný jazyk alebo určitý dialekt jazyka a jeden výstupný jazyk.

Keďže existuje veľa jazykov, existuje aj veľa migračných nástrojov medzi rôznymi jazykmi. Väčšinou ide o priamy prevod zdrojového kódu z jedného jazyka do druhého, pričom je podporovaný len jeden vstupný a jeden výstupný jazyk. Rozšíriteľnosť o ďalšie jazyky je u priamej konverzie obmedzená. Tieto migračné nástroje ale majú prístup k informáciám, ako sú názvy premenných a funkcií, komentáre a pod. Tieto informácie zachovávajú aj vo svojom výstupe, čo je ich výhodou. Neexistuje ale nástroj, ktorý by súčasne podporoval viacero vstupných a viacero výstupných jazykov.

3 MIGRAČNÝ NÁSTROJ ZALOŽENÝ NA DEKOMPILÁCII

Cieľom tejto práce je vytvoriť univerzálny nástroj podporujúci viacero vstupných a viacero výstupných jazykov. To je dosiahnuté s využitím existujúceho dekompilátoru projektu Lissom [2], ktorý slúži ako nástroj na spätný preklad. Myšlienka je podrobnejšie popísaná v [3]. Vyvinutý migračný nástroj integruje prostrednú optimalizačnú a zadnú časť dekompilátoru s prekladačmi rôznych jazykov. Prekladače zo vstupného zdrojového kódu generujú kód v LLVM IR, ktorý je vstupom prostrednej optimalizačnej časti dekompilátoru. Zadná časť dekompilátoru momentálne produkuje výstup v dvoch vysoko úrovňových jazykoch: C a Python obohatený o niektoré konštrukcie jazyka C. Takýto nástroj súčasne podporuje viacero vstupných a viacero výstupných jazykov. Pri dekompilácii sú však stratené niektoré informácie, ako napríklad komentáre, ktoré sa potom vo výstupoch nenachádzajú. Toto je nevýhodou popisovaného riešenia.

3.1 VÝBER VSTUPNÝCH JAZYKOV

Vstupné jazyky boli vyberané na základe existencie ich prekladačov, ktoré generujú LLVM IR na svojom výstupe. Zvažované jazyky boli Pascal, D, Lua, C/C++, Objective C/Objective C++, Fortran a ADA. Jazyky Pascal a Lua boli vyradené kvôli tomu, že dané prekladače už nie sú ďalej vyvíjané. O preklad jazykov Fortran a ADA sa stará prekladač GCC s pluginom Dragonegg. Bohužiaľ, plugin momentálne nepodporuje plne jazyk ADA (preklad nebolo možné ani spojazdniť), a tak bol tento jazyk taktiež vyradený. Vybrané jazyky teda sú D (prekladač LDC), C/C++, Objective C/Objective C++ (prekladač Clang) a Fortran (Prekladač GCC a plugin Dragonegg).

3.2 VYLEPŠOVANIE VÝSTUPU

Samotná integrácia prekladačov a dekompilátoru pre kvalitný výstup nie je ani zďaleka dostatočná. Výstupy sú často nepreložiteľné a veľmi nečitateľné. Aby boli použiteľné, je ich nutné značne vylepšiť. To je dosiahnuté v optimalizačnej časti vlastnými optimalizáciami pre migráciu. Tieto optimalizácie tvoria jadro migračného nástroja. Sú implementované pomocou LLVM Pass Frameworku, ktorý poskytuje nástroje na tvorbu analýz, transformácií a optimalizácií LLVM IR kódu. Implementované optimalizácie vyhľadávajú a odstraňujú nepotrebné časti kódu, definície knižničných funkcií, prevádzajú volania funkcií knižnice gfortran na volania štandardných knižničných funkcií jazyka C a pod. Príklad volania knižničnej funkcie knižnice gfortran je na prvom riadku obrázku 1. Na druhom riadku je ekvivalent zo štandardnej knižnice jazyka C. Rozdiel v čitateľnosti je znateľný na prvý pohľad. Ide len o jeden príklad z mnohých. Optimalizácie vykonané nad celým zdrojovým kódom prinášajú podstatné zlepšenie kvality výstupu.

```
((void (*)(int32_t *))_gfortran_exit_i4)(&x);
exit(x);
```

Obr. 1: Volanie knižnice gfortran

4 EXPERIMENTÁLNE VÝSLEDKY

Migračný nástroj v súčasnej dobe podporuje na vstupe jazyky Fortran, C/C++, D, Objective C/Objective C++ a na výstupe jazyky C a obohatený Python. Pre každý jazyk boli vytvorené dve sady testov. Prvá sada testuje funkčnosť migrácie a nehľadí na kvalitu výstupu, druhá sada testuje kvalitu výstupu. Každý testovací vstup z druhej sady je zameraný na špecifickú konštrukciu vstupného jazyka.

Príklad migrácie jednoduchého programu z jazyka Fortran do jazyka C je uvedený na obrázku 2. Program počíta faktoriál čísla 10 a výsledok vracia ako návratový kód pri ukončení.

Vstupný kód v jazyku Fortran

```
recursive integer function &
factorial(a) result(res)
 implicit none
 integer, intent(in) :: a
 if (a <= 0) then
  res = 1
 else
  res = a * factorial(a - 1)
 end if
end function factorial

program test
 implicit none
 integer :: a, b
 integer :: factorial
 a = 10
 b = factorial(a)
 call EXIT(b)
end program test
```

Výsledný kód v jazyku C

```
int32_t factorial_(int32_t *a1) {
 uint32_t v1 = *a1;
 int32_t v2;
 if (v1 >= 1) {
  int32_t v3 = v1 - 1;
  v2 = factorial_(&v3) * v1;
 } else {
  v2 = 1;
 }
 return v2;
}

int main(int argc, char **argv) {
 int32_t v1 = 10;
 int32_t v2 = factorial_(&v1);
 exit(v2);
}
```

Obr. 2: Ukážka migrácie z jazyka Fortran do jazyka C

5 ZÁVER

Výstupom práce je migračný nástroj postavený na prostrednej a zadnej časti dekompilátoru projektu Lissom a vybraných prekladačoch. Jadro migračného nástroja tvoria optimalizácie zvyšujúce kvalitu a čitateľnosť výstupu. Tieto optimalizácie sú hlavným prínosom práce, pretože transformujú neprelo-žiteľný a nečitateľný kód na preložiteľný kód s lepšou čitateľnosťou. Prekladače zo vstupného jazyka generujú kód v LLVM IR, ktorý je ďalej optimalizovaný a dekompilovaný prostrednou a zadnou ča-sťou dekompilátoru. Podporovanými vstupnými jazykmi sú Fortran, C/C++, D, Objective C/Objective C++. Výstupom dekompilácie je zdrojový kód v jazyku C alebo v jazyku Python rozšírenom o niektoré konštrukcie jazyka C. Migračný nástroj je jednoducho rozšíriteľný o ďalšie vstupné jazyky pridaním príslušných prekladačov generujúcich LLVM IR.

6 POĎAKOVANIE

Tento príspevok vznikol za podpory grantu FIT-S-14-2299 – Výzkum pokročilých metod ICT a jejich aplikace.

LITERATÚRA

[1] Dickens, T. Migrating Legacy Engineering Applications to Java. In *OOPSLA 2002 Practitioners Reports*. New York, NY, USA: ACM, 2002. OOPSLA'02. ISBN 1-58113-471-1.

[2] Ďurfina, L., Křoustek, J., Zemek, P. et al. Design of a Retargetable Decompiler for a Static Platform-Independent Malware Analysis. *International Journal of Security and Its Applications*. 2011, roč. 5, č. 4. s. 91–106.

[3] Ďurfina, L., Křoustek, J. a Zemek, P. Generic Source Code Migration Using Decompilation. In *10th Annual Industrial Simulation Conference (ISC'2012)*. [b.m.]: EUROSIS, 2012. s. 38–42. ISBN 978-90-77381-71-7.