Tézisfüzet

Metaprogramok alkalmazása erősen típusos objektum-orientált rendszerek kiterjesztésére

Zólyomi István

Témavezető: Dr. Porkoláb Zoltán

Eötvös Loránd Tudományegyetem
Informatika Kar
Programozási Nyelvek és Fordítóprogramok Tanszék

ELTE Informatika Doktori Iskola
Az informatika alapjai és módszertana doktori program

Iskola- és programvezető: Dr. Demetrovics János

Budapest
2009

1. Bevezetés

Napjaink legszélesebb körben alkalmazott programozási módszertana vitathatatlanul az objektum-orientált programozás. Alkalmazásának évtizedei alatt azonban világossá váltak módszertanának korlátai, hátrányai is. Ezek számos objektum-orientáltságot kiegészítő, vagy alapjaiban új paradigma létrejöttét eredményezték, ilyenek többek között az aspektus-orientált, jellemző-orientált (feature-oriented), generikus, vagy a szándékalapú programozás. Ezek számos előnyük mellett természetesen hátrányokkal is rendelkeznek. A hátrányok leküzdését és az előnyös tulajdonságok egyesítését célozta meg a többparadigmás programozás, mely a feladatnak leginkább megfelelő módszereket próbálja ötvözni.

Legígéretesebb módszertannak azonban a metaprogramozás tűnik, mely mindezen nyelvi kiterjesztések, programozási elvek absztrakciójaként értelmezhető. Alapelve nem más programozási paradigma kiterjesztésén alapszik, hanem általános programtranszformációk elvégzésére képes, a hagyományos programoktól elkülönülő metaprogramok végrehajtásával dolgozik. Ennek speciális eseteként adódik a felsorolt módszertanok mindegyike.

A programtranszformációk elvégzéséhez a metaprogramoknak hozzá kell férnie az átalakítandó program metaadataihoz, például típusinformációkhoz, kifejezésfákhoz. Másrészt képesnek kell lennie átalakításokat végezni a programon, például új kódot generálni vagy átalakítani a szintaxisfát. Sajnos a gyakorlatban a metaprogramozás még meglehetősen nehézkes és rosszul támogatott, így többnyire speciális alkalmazási területeken használják, ahol más módszerek már nem lennének célravezetők. Egyik legfontosabb a programkód tár- és időigényének optimalizálása, mely a metaprogramozás egyik első célterülete volt, és komoly eredményekre vezetett. Használják beágyazott és többszintű programnyelvek kiértékelésére, segítségükkel spe-

ciális célterületeket leíró programnyelvek illeszthetők általános célú nyelvekbe. Másik gyakori alkalmazása automatikus adattároló vagy formátumkonverziós programkód generálása típusinformációk alapján, illetve különböző protokollok szerinti automatizált kommunikációra képes kód generálása függvényszignatúrák alapján. A modern, integrált fejlesztőkörnyezetek legtöbbje képes a programkód bizonyos fokú automatizált átszervezésére is, ilyen például programkód kiemelése függvénybe, függvényparaméterek átalakítása vagy a nem elérhető kódrészletek megtalálása.

2. A dolgozat célja, módszerei

A dolgozatban az erősen típusos objektum-orientált típusrendszerek lehetőségeinek határait, alkalmazásának korlátait vizsgálom. A dolgozat célja ezen típusrendszerek lehetőségeinek kiterjesztése, melyre a metaprogramok kiváló eszközt adnak. A metaprogramok megírására a C++ nyelvet választottam, ezen belül is legtöbbször a nyelv sablonjainak segítségével dolgoztam. A sablonok a C++ nyelven belül önálló, Turing-teljes funkcionális nyelvet adnak, valamint típusparamétereik segítségével testre szabható kódgenerálási lehetőséget biztosítanak. A C++ sablon-metaprogramozás használata napjainkra elfogadott és egyre szélesebb körben alkalmazott módszerré nőtte ki magát, ezért választása természetesen adódik.

3. A dolgozat eredményei

3.1. Típusvizsgálat

Mivel a metaprogramozás egy új, feltörekvő módszertan, így még nem rendelkezik kellőképpen kiforrott elméleti háttérrel és fejlesztőeszközökkel. A programok metaadatainak elérése a metaprogramozás alapvető szükséglete, ám alig akad olyan programozási nyelv vagy környezet, mely ezt a fordítási idejű metaprogramok esetén is támogatná.

A dolgozatban bemutattam egy általános típusvizsgáló rendszert, mely lehetővé teszi az elsőrendű logika predikátumaira alapozott elemi vizsgálatok végrehajtását, valamint ezekből logikai műveletek segítségével összetett feltételeket is képes megfogalmazni. Szabványos C++ nyelvű sablon-metaprogramok segítségével megadtam a rendszer megvalósítását C++ nyelvű típusvizsgálatok elvégzésére . A témával kapcsolatos eredményeket [1, 9] alatt publikáltam.

1. Tézis. Definiáltam egy elsőrendű logikán alapuló, nem intruzív, univerzális típusvizsgáló (introspection) rendszert. C++ sablon-metaprogramok segítségével elkészítettem a rendszer egy konkrét megvalósítását, mely C++ nyelvű programkód típusvizsgálatára szolgál. A könyvtár az ISO/IEC 14882:1998 szabvány szerinti nyelvi eszközökre épül, ezért fordítófüggetlen és hordozható.

3.2. Strukturális altípusosság

A metaprogramozás fontos alkalmazási területe a nyelvek típusrendszerének kiterjesztése, tulajdonságainak javítása. A dolgozatban bemutattam az erősen típusos objektum-orientált nyelvek típusrendszerének gyengeségét a lépésenkénti finomítás módszerének alkalmazása esetén. Az objektum-orientált módszer a megoldás érdekében létrehozandó összetett osztályok, illetve interfészek számának exponenciális növekedéséhez vezet az ősök és interfészek számának függvényében.

Mivel ez a gyengeség a struktúrális altípusosság szabályainak alkalmazásával kiküszöbölhető, C++ nyelven olyan automatikus konverziókat valósítottam meg, melyek a nyelv típusrendszerének kiterjesztésével a strukturális altípusosságot szimulálják. A megoldás sablon-metaprogramokkal automatizált kódgenerálásra épül, és kizárólag szabványos nyelvi eszközöket használ. A lusta, igény szerinti kódgenerálás segítségével az újonnan létrejövő osztályok száma lineáris lett a programban ténylegesen felhasznált, eltérő konverziók számának függvényében. A megoldást [3, 5, 7] alatt ismertettem.

2. Tézis. Megmutattam a jelenlegi objektum-orientált nyelvek típusrendszerének korlátait osztályok lépésenkénti finomításának esetében. Szabványos sablon-metaprogramok segítségével a strukturális altípusosságon alapuló, kiegészítő konverziós szabályokat vezettem be a C++ nyelv típusrendszerébe, megoldást adva a lépésenkénti finomítás problémájára. A módszerrel elkerülhető a létrehozandó interfészek számának kombinatorikus robbanása.

3.3. Sorosítás

A metaprogramozás egy másik fontos alkalmazási területe az adatok sorosítása, melyre a típusok önleírásának felhasználásából kiindulva egy adott nyelvhez viszonylag jó, általános, formátumfüggetlen megoldás adható. A típusok önleírása a virtuális gépen, illetve értelmezőn alapuló programozási környezetekben (pl. Java, C#) legtöbbször adottság, ezek többnyire rendelkeznek is szabványos sorosító könyvtárakkal. Az ilyen könyvtárak közös jellemzője, hogy a nyelven megadott típusok alapján határozzák meg az elmentett adatok formátumát.

A sorosítás más, a metaadatok elérését futási időben sem támogató nyelveken jóval nehezebb. A dolgozatban bemutatott megoldás a másik irányból indul ki, és a rendelkezésre álló adatleírásokhoz, sémákhoz készít programnyelvi típusokat. A megoldás nyelv- és formátumfüggetlen, ám a formátumot leíró séma típusainak leképzését igényli a programozási nyelv típusaira. Bemutattam a sorosítás működését XML formátumú adatokra, az algoritmust a típusok önleírását nem támogató C++ nyelven megvalósítva. A megoldás hasznosságát tovább növeli, hogy kis erőforrásigénye miatt alkalmazása ideális lehet telefonok, kéziszámítógépek és egyéb, kisebb kapacitású rendszerek egymás közötti kommunikációjára. A megoldás részletei [2, 4] alatt érhetők el.

3. Tézis. Nyelv- és formátumfüggetlen, moduláris módszert adtam sémaleírással rendelkező dokumentumok sorosítására. Leképzést definiáltam az XML sémaleírók típusrendszeréről a C++ nyelv típusrendszerére, valamint megvalósítottam egy leképzést elvégző kódgenerátort. Implementáltam a leképzésre épülő, XML dokumentumokat sorosító általános metaprogramot. Az elkészült könyvtárat a Nokia hivatalos fejlesztői eszközként adta ki S60 platformjára.

4. További kutatási lehetőségek

Természetesen a dolgozat megoldásainak bármelyike adhatna lehetőséget további fejlesztésekre, azonban én mégsem ezt érzem a legfontosabb kutatási iránynak. A kutatási munka során minduntalan olyan nehézségekbe, korlátokba ütköztünk, melyek messze nem a metaprogramozás határainak eléréséből, hanem annak gyermekbetegségeiből adódnak. Azt gondolom tehát, hogy a metaprogramozás vizsgálata és fejlesztése, módszereinek megalapozása lehet a további kutatások legfontosabb iránya.

A metaprogramozás leggyakrabban még mindig ad-hoc jellegű megoldásokra alapul, súlyos módszertani hiányosságokkal rendelkezik, ami a megfelelő elméleti alapok, illetve a megfelelő programozási eszközök hiányának tudható be. Fontosnak tartom tehát megtalálni azokat az elméleteket, módszertanokat, melyek segítségével jól tervezett, helyes és karbantartható metaprogramok írhatók. Másrészt a hagyományos programok fejlesztésében bevált eszközök jó része hiányzik a metaprogramok esetében. Nincsenek bonyolultságot mérő metrikáink, tesztkörnyezeteink, nyomkövetést vagy teljesítménymérést biztosító eszközeink, integrált fejlesztőkörnyezeteink sem. Fontosnak tartom tehát metaprogramozást támogató fejlesztőeszközök elkészítését is.

Hivatkozások

[1] István Zólyomi, Zoltán Porkoláb. Towards a General Template Introspection Library. Generative Programming and Component Engineering LNCS Vol. 3286 (2004) pp. 266-282.

[2] Szabolcs Payrits, Péter Dornbach, István Zólyomi. Metadata-Based XML Serialization for Embedded C++. Proceedings of ICWS 2006, pp. 347-356

[3] István Zólyomi, Zoltán Porkoláb, Tamás Kozsik. An Extension to the Subtype Relationship in C++ Implemented with Template Metaprogramming. Generative Programming and Component Engineering LNCS Vol. 2830 (2003) pp. 209-227.

[4] Szabolcs Payrits, Péter Dornbach, István Zólyomi. Metadata-Based XML Serialization for Embedded C++. International Journal of Web Services Research (IJWSR), megjelenés alatt.

[5] István Zólyomi, Zoltán Porkoláb. A Feature Composition Problem and a Solution Based on C++ Template Metaprogramming. Generative and Transformational Techniques in Software Engineering LNCS Vol. 4143 (2006) pp. 459-470.

[6] István Zólyomi, Zoltán Porkoláb. A generative approach for family polymorphism in C++. Proceedings of ICAI 2004, Eger, Hungary, pp. 445-454.

[7] Zoltán Porkoláb, István Zólyomi. An anomaly of subtype relations at component refinement, and a generative solution in C++. MPOOL Workshop, ECOOP 2004, Oslo, pp. 39-44.

[8] Ádám Sipos, István Zólyomi, Zoltán Porkoláb. On the Correctness of Template Metaprograms. Proceedings of ICAI 2007, Eger, Hungary, pp. 301-308.

[9] István Zólyomi, Zoltán Porkoláb. Improving concept checking in Boost. Boost Workshop, OOPSLA 2004, Vancouver.