DOTAZOVÁNÍ NAD STROMEM ABSTRAKTNÍ SYNTAXE

Josef Smolka, Miroslav Virius
Fakulta jaderná a fyzikálně inženýrská ČVUT v Praze
smolkjos@fjfi.cvut.cz

ABSTRAKT:
Příspěvek se zabývá problematikou vyhledávání informací ve stromě abstraktní syntaxe, a to především z pohledu implementace refaktorovacích a analytických nástrojů. Jako možné řešení představuje jazyk XQuery, který byl původně navržen jako dotazovací jazyk pro XML databáze, a návrh mezivrstvy mezi stromem abstraktní syntaxe a jazykem XQuery.

ABSTRACT:
The purpose of this paper is to cover the problem of retrieving the desired information from the abstract syntax tree (AST), particularly in terms of implementation of refactoring and source code analysis tools. The XQuery language, which was originally designed as a query language for XML databases, is proposed as possible solution. Also a design of interlayer between XQuery language and abstract syntax tree is presented.

KLÍČOVÁ SLOVA:
strom abstraktní syntaxe, Java, refaktorování, analýza kódu, XML, XPath, XQuery

ÚVOD

Strom abstraktní syntaxe představuje časem prověřenou volbu reprezentace programu pro potřeby dalšího strojového zpracování. Zachovává strukturu programu při vynechání některých syntaktických detailů, které nejsou pro další zpracování relevantní, a je dobrým výchozím bodem pro tvorbu obecnějších struktur zachycujících jednotlivé vazby mezi částmi programu. Nejedná se však o nijak jednoduchou strukturu, například syntaktický strom jazyka Java rozlišuje přes sto různých typů uzlů a může narůst do takové hloubky a šíře, že jeho zpracování rekurzivními algoritmy by mohlo narazit na omezenou velikost zásobníku. To nastoluje otázku, jak jednoduše (tedy ne nutně efektivně) získat z takovéto struktury požadované informace.
Tento příspěvek ukazuje problémy, které bylo třeba řešit při využití stromu abstraktní syntaxe jako podkladové struktury pro implementaci nástroje pro refaktorování a dekompilaci programů napsaných v jazyce Java.

SOUČASNÝ STAV

V současné době existuje několik nástrojů a knihoven, které umožňují převod stromu abstraktní syntaxe do jazyka XML, nejsou však využívány a dosud neexistuje ani standard pro vyjádření struktury programu v jazyce XML [3], [5]. Algoritmy, které využívají dotazovací jazyk XQuery při práci se stromem abstraktní syntaxe v XML při dekompilaci nebo při refaktoringu, dosud nebyly, jak se zdá publikovány. Zdá se, že ani neexistuje programový produkt, který by na nich byl založen.

NÁVRHOVÝ VZOR NÁVŠTĚVNÍK

První možností zpracování stromu abstraktní syntaxe, kterou jsme zkoumali, je návrhový vzor Návštěvník (Visitor). Podívejme se, jak ho implementuje knihovna JDT integrovaného vývojového nástroje Eclipse, která má na starosti práci s kódem. Základem řešení je

abstraktní třída `ASTVisitor`, která obsahuje deklarace následujících dvou metod pro každý typ uzlu ve stromě abstraktní syntaxe:

- `public boolean visit(T node)` – navštíví předaný uzel, a pokud je vrácena logická hodnota `true`, pokračuje v návštěvě také u následníků tohoto uzlu;
- `public boolean endVisit(T node)` – je volána po návštěvě následníků předaného uzlu.

Kromě těchto metod obsahuje třída `ASTVisitor` také dvě společné metody pro všechny typy uzlů: `preVisit`, která se volá před `visit`, a `postVisit`, která se volá po `endVisit`. Návštěva následníků probíhá vždy v takovém pořadí, které se nejvíce blíží pořadí, v jakém zdrojový kód zpracovává lexikální analyzátor [4]. Ukažme si například, jak vypsat na obrazovku názvy všech deklarovaných lokálních proměnných v dané kompilační jednotce:

```
public boolean visit(VariableDeclarationStatement vds) {
  List<VariableDeclarationFragment> frags = vds.fragments();
  for (VariableDeclarationFragment vdf : frags) {
    System.out.println(vdf.getName());
  }
  return false;
}
```

Vytvoříme třídu `MyVisitor` odvozenou od abstraktní třídy `ASTVisitor` a překryjeme metodu `visit(VariableDeclarationStatement)`, jak je naznačeno v předchozí ukázce kódu. Takto definovaného návštěvníka můžeme aplikovat na kompilační jednotku:

```
CompilationUnit cu = (CompilationUnit) parser.createAST(null);
cu.accept(new MyVisitor());
```

Nevýhodou tohoto způsobu vyhledávání různých informaci ve stromě abstraktní syntaxe je, že pro různé dotazy musíme psát nové implementace metod třídy `ASTVisitor`. Navíc implementace některých dotazů může být velice komplikovaná.

VYUŽITÍ JAZYKA XPATH

Další možností, kterou jsme zkoumali, bylo využití jednoduchého navigačního jazyka XPath. Dotazování se na informace ve stromové struktuře je ve světě XML naprosto běžná operace. K tomuto závěru došli například i tvůrci nástroje PMD, který slouží pro analýzu zdrojových kódů v Javě. Ti se rozhodli využít jazyk XML Path (XPath) pro odkazování na části stromu abstraktní syntaxe, nad kterým provádějí analýzu zdrojového kódu. Zdrojový kód je nástrojem PMD posuzován pomocí sady pravidel, které mohou být implementovány buď přímo v Javě a nebo mohou být zapsány jako výrazy v jazyce XPath. Druhá možnost bývá mnohdy jednodušší a sami autoři ji doporučují [5]. Například pro získání lokálních proměnných v dané kompilační jednotce, podobně jako v předchozím příkladě, stačí v PMD takto jednoduchý XPath výraz: `//LocalVariableDeclaration`
I toto řešení má některé nevýhody. Jazyk XPath, jak již vyplývá z jeho názvu, je primárně určen pro jednoduché adresování konkrétních prvků ve stromě. Jakmile nastane potřeba provádět v rámci dotazu nějaké složitější rozhodování, přestane XPath stačit. Často se tedy jazyk XPath používá v rámci jiného komplexnějšího jazyka [2].

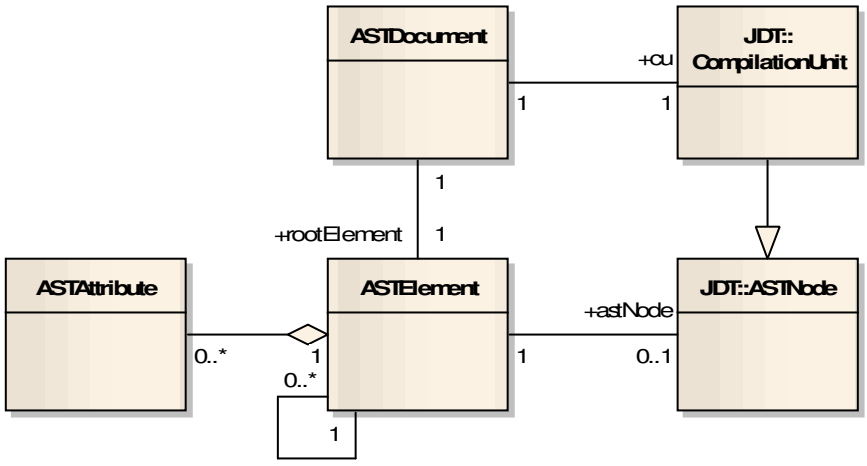
OD XPATH K XQUERY

Tímto komplexnějším jazykem by mohl být například funkcionální dotazovací jazyk XQuery. Jedná se o jazyk odvozený od původního jazyka Quilt, který vznikl jako reakce na požadavek W3C na univerzální dotazovací jazyk nad XML. Quilt se inspiroval různými dotazovacími jazyky, jako jsou např. XPath, XQL, XML-QL, SQL a OQL. XQuery verze 1.0 má společný základ s XPath verze 2.0 a tyto dva jazyky jsou velice úzce spjaty [1]. Je tedy možné v rámci dotazů v XQuery využívat výrazy XPath. Dotazy mají podobnou strukturu jako v jazyce SQL; v XQuery se tato struktura označuje zkratkou FLWOR:

```
for $book in ./library/books/book
let $author := /library/authors/author[id = $book/author]
where $book/kategorie = 'sci-fi'
order by $book/name
return ($book, $author)
```

Mezivrstva mezi XQuery a AST

Aby bylo možné použít jazyk XQuery pro dotazování nad stromem abstraktní syntaxe, museli jsme vytvořit mezivrstvu mezi interpretem jazyka a stromem abstraktní syntaxe, aby se strom abstraktní syntaxe jevil pro XQuery jako obyčejné XML. Tato mezivrstva nemněla jen slepě kopírovat strukturu syntaktického stromu, ale měla za úkol provést určité optimalizace, které mohou zjednodušit případné dotazy; to zahrnuje i rozhodnutí, jaké uzly budou nově představovat elementy a jaké budou reprezentovány pouze jako atributy těchto elementů. Pro tyto potřeby jsme vytvořili knihovnu JReflib, která dokáže převést syntaktický strom do reprezentace DOM (Document Object Model). Elementy ve stromu DOM si však stále drží odkazy na odpovídající uzly ve stromě abstraktní syntaxe, viz diagram tříd na obrázku 1.



Obrázek 1: Diagram tříd AST DOM

Implementace XQuery pro Javu

Platforma Java zatím bohužel neobsahuje oficiální implementaci jazyka XQuery – v této spojitosti se zmiňuje až Java 7. Naštěstí již nyní existuje několik kvalitních open source implementací, které je možno využít. Jako první jsme zkoumali GNU Quexo, které je postaveno na frameworku GNU Kawa určeném pro implementaci vysokoúrovňových dynamických jazyků kompilovatelných do bajtového kódu Javy. To přímo vybízí k využití XQuery v servletech. Bohužel pro účely integrace do refaktorovacího nebo analytického nástroje se příliš nehodí kvůli implementaci DOM, která se jen velice těžko rozšiřuje.

Jako druhou variantu jsme zvolili Berkeley Nux, což je open source knihovna pro práci s XML, obsahující implementaci XQuery i s experimentální aktualizační funkcí (update), dále implementaci XPath, fulltextového vyhledávání i podporu pro binární XML. Sice používá také vlastní implementaci DOM, tzv. XOM, ale tato implementace je velice snadno rozšiřitelná. Od tříd Document, Element a Attribute jsme odvodili vlastní třídy s předponou AST (viz diagram tříd na obrázku 1), ze kterých skládáme DOM reprezentaci předaného syntaktického stromu. Použití v námi implementovaném refaktorovacím nástroji vypadá například následovně:

```
ASTDocument doc = ASTDocFactory.forFile(javaFile);
XQuery xq = XQScriptFactory.forName("my-script.xq");
Nodes rs = xq.execute(doc, null, vars).toNodes();
```

Obě naše tovární třídy používají kešování dotazů, aby se předešlo zdržování v důsledku IO operací a vytváření nových objektů. Důležitou vlastností obou zmiňovaných implementací XQuery je možnost volat ze skriptu statické metody tříd, předávat jim parametry a pracovat s jejich návratovou hodnotou:

```
declare namespace rfunc = "java:my.package.XQFunctions";
rfunc:myFunction($arg1, 'arg2')
```

UKÁZKA XQUERY DOTAZŮ NAD AST

Pro ilustraci zde uvedeme dva dotazy, které využíváme při implementaci refaktorovacího nástroje pro jazyk Java.

Jména všech atributů třídy

```
for $type in .//type
where $type/name = $typeName
return $type/fields/field/name
```

Nalezení metody

```
for $type in .//type
let $methods := $type/methods/method[name = $methodName]
where $type/name = $typeName
return if (count($methods) = 1)
  then $methods[1]
  else for $m in $methods
    where rfunc:methodSignature($m) = $methodSignature
    return $m
```

ZÁVĚR

Tento příspěvek ukázal některé problémy, na které narážíme při implementaci dotazů nad stromem abstraktní syntaxe. Jako možné řešení představil využití nástrojů pro zpracování XML, především jazyků XPath a XQuery. Za tímto účelem jsme implementovali knihovnu JRefLib, která představuje mezivrstvu mezi AST a jazykem XQuery.
Poznamenejme, že jde o součást rozsáhlejšího projektu implementace refaktorovacího a dekompilačního nástroje pro Javu.

PODĚKOVÁNÍ

Práce na tomto příspěvku byla podporována z grantů MŠMT LA08015 a SGS 10/094.

LITERATURA

[1] Boag, S.; Chamberlin, D.; Fernández, M. F.; Florescu, D.; Robie, J.; Siméon, J. *XQuery 1.0: An XML Query Language*. Version 1.0. W3C Recommendation [online]. 23. 1. 2007 [cit. 2010-03-14]. Dostupný z WWW: <http://www.w3.org/TR/xquery/>.

[2] Clark, J.; Derose, S. *XML Path Language (XPath)* Version 1.0. W3C Recommendation [online]. 16. 12. 1999 [cit. 2010–03–13]. Dostupný z WWW: <http://www.w3.org/TR/xpath/>.

[3] Collado, M. *Projects related to XML representation of source code* [online]. 2009, [cit. 2010-03-13]. Dostupný z WWW: <http://lml.ls.fi.upm.es/~mcollado/emu-code/emu-code-other.html>

[4] Kuhn, T.; Thomann, O. *Abstract Syntax Tree*. Eclipse Corner Articles [online]. 20. 11. 2006, [cit. 2010–03–13]. Dostupný z WWW: <http://www.eclipse.org/articles/article.php?file=Article-JavaCodeManipulation_AST/index.html>.

[5] PMD [online]. 2009 [cit. 2010–03–14]. *XPath Rule tutorial*. Dostupné z WWW: <http://pmd.sourceforge.net/xpathruletutorial.html>.