形式手法ツール用のスクリプト言語について

岡野浩三[†1]

本稿ではソフトウェア解析に用いられる形式手法を活用したツールを開発するために役立つと思われるスクリプト言語の要件について述べる．

On Script Languages for Tools of Formal Techniques

Kozo Okano[†1]

This report describes requirements on script languages for developping tools which analyze software using formal techniques.

1. はじめに

ソフトウェア開発における形式手法は重要性を増しており，これらに関する研究は極めて多い．著者の研究グループも文献[1),2)]のように多くの研究を行ってきている．このような研究はその手法を実現するためのプロトタイプツールの作成が評価実験のために必須であるが，その開発や保守に多くの時間を要し，研究の進展の速度向上の阻害要因になりつつある．

一般に，評価実験で必要なことは，多くの手法やツールとの比較実験であり，そのためには，多種の要素技術を切り替えながら，さまざまなパラメータでの評価が必要になる．この観点からは，Java でコンパイルして評価用ツールを実装するよりは，さまざまな要素技術を容易に組み合わせて多くの手法を抽象度の高いレベルで記述・実行できるようなスクリプト言語の活用が望まれる．

そこで本稿では，具体的なプロトタイプツールを元にし，そのようなスクリプト言語に望まれる要件を整理する．

2. 対象ツール

本稿で対象にするツールは，著者らの論文[1)]で作成したツールである．このツールでは以下のような要素技術が用いられている．

(1) 表明構文解析
(2) プログラム解析のための JDT
(3) SMT ソルバー Z3
(4) プログラム実行パス解析のためのライブラリ masu

さまざまな評価実験を行うためには，例えば Z3 を他の SMT ソルバに置き換える，別の実行パス解析のツールに置き換える，プログラムスライス解析の処理を加える，表明構文を変更する，解析の一部の処理，あるいは全部の処理を動的解析ツール，統計解析ツールで置き換えるなどの操作が考えられる．

3. 動機・目的

これらの処理をするたびに Java のプログラムを変更し，コンパイルするのは手間がかかり，また，本質でないところでプログラム開発の手間がかかる．

また開発されたプログラムの保守を研究室の学生で行っていくことが難しくなる．

このような問題点をある程度解決できるアプローチとしてこのようなツールの本質的な部分をスクリプト言語で記述することが考えられる．適切な API の整備のもとで適切なスクリプト言語を選択・あるいは設計することで，処理の本質的な部分を抽象度の高いレベルで記述でき，そのスクリプト自体を可読性の高い処理のドキュメント文章として管理でき，保守や評価実験の記録に用いることができる．

4. 具体的要件

現在，研究グループの多くの評価ツールは Java ベースで作成している．このため，既存のツールや開発ライブラリの効率の良い再利用が必須と考える．

そのためには，スクリプト言語は Java ライブラリや

†1 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University

クラスファイルの呼び出しができることが前提となる．

この観点から本質的な要件をまとめると次のようになる．

(1) Java の既存ライブラリ，クラスファイルの呼び出しが可能であること
(2) 適切なモジュール構造，あるいはオブジェクト指向を持っていること
(3) 外部 API を持っていない単体のアプリケーションの呼び出しと実行結果の取得が容易なこと
(4) API の作成，拡張が容易なこと

上記の要件を満たすスクリプト言語は数多いと思われるが，そのスクリプト言語の開発ドキュメントの多さなどを考慮に入れると，例えば Jython[3], Scala など JVM 上で動くスクリプト言語が有力となると思われる．また上記の 3 番目の項目を考慮に入れると GUI ベースのアプリケーションでも自動操作できる Python のライブラリを持つ Jython の利用が有力と考えられる．

次に具体的な API の一部を Java の API 表記で列挙していく．

4.1 プログラムソース解析

ClassInfo getClass(String filepath): filepath で指定された java ファイルから クラス情報をすべて含むオブジェクトを取得する．

Set<MethodInfo> getMethods(): そのクラスに属するすべてのメソッド情報の集合を返す．

Set<VarInfo> getLocalVariables(): そのメソッドに属するすべてのローカル変数の情報の集合を返す．

List<StatementInfo> getStatements(): そのメソッドのボディの文リストを返す．

PDG getPDG(): そのメソッドの PDG を返す．

4.2 構文変換関係

List<StatementInfo> TranslateStatements(XML rules, int start, int end): そのメソッドの指定区間の文列を rules に従って構文変換を行う．

StatementInfo composit(Operator op, StatementInfo st1, StatementInfo st2): st1, st2 をオペレータ op で結合したあとの文の statementInfo を返す．

4.3 ソルバー関係

Results executeSolver(SolverType sl): その Statement に対して，指定されたソルバーで求解する．結果は Results 型のオブジェクトで返り，そのオブジェクトに対し，解や反例をさらに getter で取得できる．

4.4 動的実行関係

Results executeAnalisys(DyanamicAnalysis da, String[] args): 動的解析ツール da を 引数 args で実行し，結果を Resutls 型のオブジェクトとして得る．

4.5 主処理のプログラム記述例

Java 風の記述であるが，上記の API を用いた主処理の具体的な記述例を以下に与える．

```
Script sp = new Script();
ClassInfo clinfo =  sp.getClass("Test.java");

for (MethodInfo method: clinfo.getMethods()) {
   PDG  pdg = method.getPDG();
   MyRes an1 =  analyze(pdg);
   sl = TranslateStatements(myRules,
      method.getStatements(),
      an1.start, an1.end);
   Rersults rs = sl.executeSolver(Builtins.Z3):
   system.out.println(rs);
}
```

この記述の意図は，次のとおりである．与えられた Java クラスのソースファイルに対して，各メソッドごと PDG を作成し，得られた PDG に対して解析 analyze を実行し，そこで得られた範囲の文に対して myrule で定義された変換規則で構文変換を行った結果にし対し，Z3 を適用し，結果を出力している．

5. おわりに

そこで本稿では，具体的な形式手法評価用プロトタイプツールを元にし，そのようなツールを開発するために必要なスクリプト言語に望まれる要件を整理した．また API の具体例を列挙した．今後は API を精練し，このようなスクリプト言語を DSL として開発あるいは，既存言語のライブラリとして開発し，さらには，それを用いて実際の研究において評価ツールの実装を行っていきたい．

参 考 文 献

1) 小林和貴，佐々木幸広，岡野浩三，楠本真二: “PDG と SMT ソルバを利用した表明自動導出手法の提案と評価”，電子情報通信学会論文誌, Vol.J96D, No.11, pp.2657-2668 (2013)
2) Kentaro Hanada, Hiroaki Shimba, Kozo Okano, and Shinji Kusumoto: “Implementation of a Prototype Bi-directinal Translation Tool between Ocl and Jml,” International Journal of Informatics Society, Vol.5, No.2, pp.89-96 (2013)
3) Jython http://www.jython.org/