形式仕様記述言語 Alloy からの C#コード生成

大分高専電気電子情報工学専攻　　渡邊優樹　　（指導教員　西村俊二）

１．研究背景

近年，ソフトウェア開発において，自然言語によって記述された仕様の曖昧さによる発生するバグを回避するために，形式手法と呼ばれるプログラム開発手法が注目を集めている．

形式手法の１つに，形式仕様記述言語 Alloy を用いる手法がある．Alloy を扱うツールとして Alloy Analyzer があるが，Alloy で記述された仕様から他言語のプログラム･コードを生成する機能は備わっていない．これまで，Alloy から JML 仕様の生成の研究[1]等は行われているが，ソフトウェア開発において用いられる事の多い C#等のコードを Alloy から直接生成できれば，プログラムの開発効率もあがり，ヒューマンエラーの予防も可能であると考えらる．

２．目的

本研究では，Alloy から C#コードを生成する機能がないことに着目し，Alloy で記述された仕様のうち，ソフトウェアの実装に関わる部分から C#コードを生成する手法の提案と開発を行う．

３．生成方法

3.1　概要

Alloy からの C#コード生成は，おおまかに以下の手順で行う．

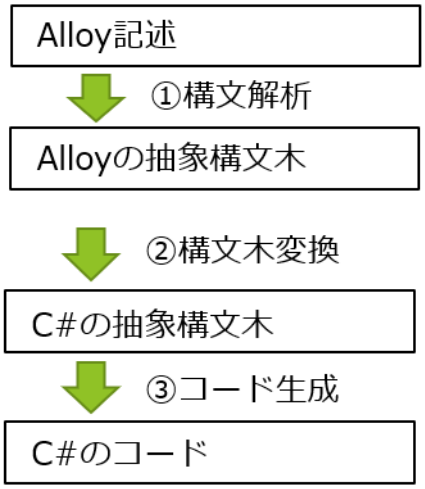

図１．C#コード生成方法

3.2　Alloy 記述の構文解析

Alloy 記述のうち，assert や check といった変換する必要のない物を除いた部分から Alloy の抽象構文木を生成する処理である．Alloy Analyzer に Alloy 記述を解析する機能は備わっているが，出力できないため，他の言語で扱えるファイルとして出力する機能の実装を行う．

3.3　C#の抽象構文木の生成

Alloy の抽象構文木から C#の抽象構文木へ変換する処理である．Alloy のコードのうち，C#コードへの変換の際に必要のない部分の除去を行う．3.5 で詳しく説明する．

3.4　C#コードの生成

C#の抽象構文木から C#コードを生成する処理である．.NET コンパイラプラットフォーム Roslyn を用いて C#で実装する．

3.5　抽象構文木の変換

Alloy の各要素毎に表１のように，対応する C#の要素へと変換する．

表１．変換対応表

Alloy	C#
sig	クラス
pred	メソッド
fun	メソッド
parameter	引数
field	フィールド
variable	フィールド

概ね表１の通りに変換するが，pred に関しては，プログラムの処理ではなく制約を示している場合があり，その場合はメソッドには変換しない．

各要素には論理式や文が含まれ，全て C#の文へ変換する．変換について，一部を表２に示す．

表２．変換対応表

Alloy	C#
open	using
extends	extends
call	関数呼び出し
set	Hashset
->	Dictionary
a => b	if(a){b}

表には載せていないが，基本的な四則演算や比較に関しては手を加えず，そのまま C#コードでも使用する．

４．進捗状況

一部手動で行う必要があるが，Alloy 仕様から Alloy の抽象構文木の生成，出力に成功した．また，簡単な C#コードの生成を行った．

５．今後の予定

プログラムの制約を表現する fact の変換について検討する必要がある．また，変換可能なプログラムパターンの増加も行う．

参考文献

1) Daniel Grunwald, Christoph Gladisch, Tianhai Liu, Mana Taghdiri, and Shumuel Tyszberowicz：Generating JML Specifications from Alloy Expressions，2014