在庫管理プログラムに対する Alloy Analyzer を用いた検証事例

森　恵弥佳[†1]　岡野浩三[†1]　楠本真二[†1]

本稿では，共通問題である在庫管理プログラムの仕様を Alloy のモデルとして記述したケーススタディについて報告する．モデル記述にあたっては，事前に記述していた JML による契約をリファレンスとした．Alloy Analyzer での検証とともに，JML による記述との比較を行い，Alloy の記述，Alloy Analyzer のモデルファインダーとしての能力についてこの事例研究を通じての有効性を示す．

A Case study — Verification of Warehouse Management Program by Alloy Analyzer —

Emika Mori,[†1] Kozo Okano[†1] and Shinji Kusumoto [†1]

This report describes a case study in which a ware-house program, the famous common problem for software design, is analyzed by Alloy analyzer. We use JML annotations which have been already described in our previous study, as the reference of specification. Comparing the result with the JML annotation, we show the power of Alloy analyzer, a tool for detecting defects in an abstract level of specification.

1. はじめに

ソフトウェア開発手法のひとつとして，契約による設計（Design by Contract）[1] と呼ばれる手法がある．各クラス・メソッドが満たすべき仕様を明確に定義することで，ソフトウェアの仕様理解の補助や，プログラム検証を行える．

DbC を実現するための具体的な言語として，Java のソースコード中に契約を記述可能な JML (Java Modeling Language)[2] などが存在する．Alloy[3] は自動解析ツールを備えており，記述したモデルから契約の検証を行える．本研究では，共通問題である在庫管理プログラム[4] の仕様を Alloy で記述し，Alloy Analyzer で検証を行った．その結果，以前著者らの研究グループが作成した契約の誤りが発見された．この誤りに対する修正記述も本稿で提示する．この結果から Alloy による検証の有用性を確認できた．

2. Alloy

Alloy は Z 言語をもとにした一階関係論理に基づく形式仕様言語である[5]．自動解析ツール Alloy Analyzer を備えており，ユーザは Alloy でモデルを記述することで，そのモデルの性質を自動で解析でき，充足例や反例などの視覚的なフィードバックを得ることができる．実行コードやテストケースなどは必要なく，専門家によるものよりも，すばやく，正確なレビューが可能となる[6]．Alloy を活用すれば，実装前の段階でモデルの欠陥を発見することができ，実装での手戻りのコストの削減につながる．

3. モデルの記述

Alloy を用いて契約の正しさを検証するために，従前研究で著者らの研究グループが付加した JML だけでなく，メソッド自体も Alloy の制約 (論理式) で記述した．

以下に，モデルの記述方法について簡単に説明する．まず，各クラスはシグネチャとして記述した．フィールドについては，String 型と，識別子として利用されている Integer 型を，Alloy に標準で備わっている Int や String ではなく，識別子としての性質を考慮し，独自に定義したシグネチャを用いて表現した．また，リストは集合として扱った[5]．

コンストラクタとメソッドに関しては，副作用のないもので，かつ内包表記を使用せずに関数として記述できるものは関数として記述し，そうでないものは制約として記述した．それらの関数・制約の引数は，オブジェクトの事前状態・オブジェクトの事後状態（副

†1 大学院情報科学研究科/基礎工学部
Graduate School of IST/ School of Engineering Science, Osaka University

```
pred shippingItem_org[c,c':ContainerItem, n:Str,a:Int,r:Int] {
        inv[c] && shippingItem_pre[n,a]  && c.shippingItem[c',n,a,r]
}
assert shippingItem_org1 {
        all c,c':ContainerItem, n:Str,a:Int, r:Int |
                shippingItem_org[c,c',n,a,r]
                        => shippingItem_post_org[c,n,a,r]
}
```

図 1　shippingItem() の検査のための Alloy 記述
Fig. 1　Alloy Description for Verification of shippingItem()

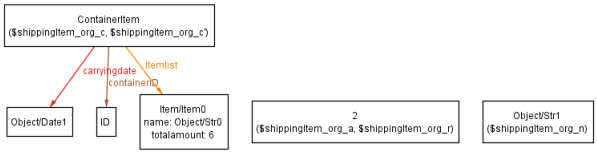

図 2　shippingItem() の事後条件を満たさない例
Fig. 2　Counterexample to the postcondition of shippingItem()

作用のあるメソッドの場合)・引数・戻り値とし，それらの間に成り立つ論理式をモデルとして記述した．戻り値がリストであるメソッドなどにおいて高級限量エラーのため解析できない場合があったが，戻り値を除いた場合の記述で代用して解析を行った．

メソッドの実行は，“事前状態での不変条件の成立とメソッドの事前条件の成立，そして制約として記述したメソッドの成立の論理積”で表す．契約の検査は，“メソッドの実行が，事後状態での不変条件の成立とメソッドの事後条件の成立を含意する”の反例を探すことで行う．また，メソッドの実行が真になるインスタンスを探すことでメソッドの実行例が得られる．

注意すべき点は，もしメソッドの記述が誤っておりメソッドの実行を満たすインスタンスが存在しなかった場合は，検査の制約中にある含意の左項が恒偽となり，反例が存在しないことである．よって，契約の検査では，反例がないことだけでなくメソッドの実行例が存在することも確認しなければならない．

4. 検 証 結 果

検証結果の具体例を示す．ContainerItem クラスのメソッド shippingItem() は引数で指定された品名と数量の要求をできるだけ満たすようコンテナから取り出す．戻り値は不足量である．“コンテナ内には引数で指定された品名の品目が含まれておりかつ戻り値は要求量からメソッド実行前の状態での所有量を引いたものである”という事後条件が記述されていた．この検証のための記述の一部を**図 1** に示す．スコープ 3 で検証を行うと，**図 2** のような反例が出力される．この反例は，“引数で指定された品名の品目がコンテナ内に存在しないため，事後条件が成立しない”という例である．正しい事後条件は，“コンテナが事前状態で指定された品名の品目を所有しているのであれば，コンテナ内には引数で指定された品名の品目が含まれておりかつ戻り値は要求量からメソッド実行前の状態での所有量を引いたものであり，所有していないのであれば戻り値は要求量と等しい”である．事後条件を適切と思われるものに書き換え，スコープ 20 で検証した結果，“No counterexample found. Assertion may be valid.”と出力され，この事後条件はスコープの範囲において正しいことがわかった．以上のように契約の誤りを見つけ，正しい契約に修正することができた．

5. 今後の予定

今後は，今回使用した在庫管理プログラムを Alloy の記述方法を変えて，本稿のものと発見できる誤りの違いや，時間や労力の差などを比較することを考えている．また，さらに別のプログラムで同様に検証を行い，Alloy Analyzer を用いた別の実例を示すことも考えている．

6. お わ り に

本稿では，Java で記述された在庫管理プログラムの仕様を Alloy のモデルとして記述した．そして，作成したモデルをもとに Alloy Analyzer でプログラムにあらかじめ付与されていた契約を検証した．検証を通じて，Alloy の記述，Alloy Analyzer のモデルファインダーとしての能力について有効性を示した．

参 考 文 献

1) Meyer, B.: *Object-oriented software construction (2nd ed.)*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1997).
2) JML: The Java Modeling Language, `http://www.eecs.ucf.edu/~leavens/JML/index.shtml`.
3) Alloy: `http://alloy.mit.edu`.
4) 尾鷲方志，岡野浩三，楠本真二：JML を用いた在庫管理プログラムの設計と ESC/Java2 を用いた検証，電子情報通信学会技術報告，Vol.107, No.176, pp.37–42 (2007).
5) Jackson, D.: *Software Abstractions: Logic, Language, and Analysis*, The MIT Press, revised edition (2012).
6) Jackson, D.: Alloy: A Lightweight Object Modelling Notation, *ACM Transactions on Software Engineering and Methodology*, Vol.11, No.2, pp.256–290 (2002).