B-037

Zによる仕様記述と状態遷移規則の比較による誤り検出法

The Error Detecting Method by Comparing a Z Specification with State Transition Rules

平岡 雅也† 織田 健‡
Masaya Hiraoka Takeshi Oda

1. はじめに

ソフトウェア開発の生産性や品質を向上させるためには、要求仕様の誤りを減らすことが有効である。形式的仕様記述言語を用いることによって曖昧さは除かれ、矛盾は数学的に検出することが可能となる。

検証可能であり、最も広く使用されている言語として仕様記述言語Zがある。しかしZはシステムの状態遷移を操作前後の関係で記述するため、システム全体の動作が捉えづらく、必要な遷移を起こすスキーマの不足などを検証することはできない。本研究ではシステムの状態遷移を直接表わす状態遷移規則を併記し、Z仕様との比較により誤りを検出する手法を提案する。

2. 仕様の記述法

小池らはZ仕様を対象としスキーマによって起こる集合の変化と関係の変化と状態遷移規則として得た知識との比較によりスキーマの妥当性を検証する手法を提案している [1]。齋藤らは形式的仕様における操作の制約条件の欠落を自動検出しその解消の支援を可能にすることを目的とした欠落検出を提案している [2]。これらは状態遷移規則をドメイン知識として正しい規則として扱っていたが、Z仕様とドメイン知識による状態遷移規則では一致できる範囲が限られてしまう。

そのため、本手法ではZ仕様と状態遷移規則を情報交換を絶った状態で2グループにより記述する。その後それぞれの要素を対応付け比較を行うとする。

ドメイン知識ではなく同一システムの仕様同士の対応付けを行うことによりより詳細に仕様の比較を行える。またZ仕様は操作ごとの動作を記述し、状態遷移規則はシステム全体の状態遷移を記述するため混入する誤りの種類が異なると思われる。これらの特長から以前の手法より効率的に誤りを検出できると考えられる。さらに状態遷移規則の記述法を拡張しイベントに関するエラー処理を記述することで、エラー処理に関してもZ仕様との対応付けにより誤りの検出を行う。

2.1 仕様記述言語 Z

Zは集合論と述語論理に基づく仕様記述言語である。変数は型を持った集合であり、集合の要素の増減によってシステムの状態遷移を表わす。

仕様は複数のスキーマにより記述し、システムの抽象状態を記述するスキーマ (状態スキーマ) とシステムに対する操作を記述するスキーマ (操作スキーマ) に分類できる。操作スキーマでは操作前、操作後の変数の間に成り立つ関係を記述することで操作による集合の変化を表す。操作スキーマにおける述語は操作後の変数の値を規定する述語 (本手法では規定式と呼ぶ) と、操作を行なうための制約条件を示す述語 (本手法では制約式と呼ぶ) に分けられる。

	非利用	通常利用
貸出	OK	[#貸出本<貸出上限]/OK [#貸出本≦貸出上限]/ERR1
返却	ERR2	OK

図 1: 利用者の状態遷移規則

2.2 状態遷移規則

状態遷移規則はエンティティの種類ごとに存在し、状態、イベント、イベントによって起こされる状態遷移からなる。状態遷移規則はエンティティの状態遷移を直接表わすためシステム全体の動作をつかみやすい。

状態遷移は遷移条件によって制限される。本手法では遷移条件をZ記法により複数の変数 (条件変数) からなる論理式として記述する。またエラー処理を考慮し、エンティティの種類ごとに状態とイベントの組合わせで起こるエラー処理を表形式で記述する。正常なイベントが起きる場合はOKを返すとする。

図1に図書館システムにおけるエンティティである利用者の状態遷移規則を示す。下部は利用者の状態遷移のエラー処理表となる。イベントに対して起こりうるエラー処理は複数あるが、利用者の状態に関わるエラー処理のみ記述する。

3. 誤り検出法

本手法ではエンティティはZ仕様では何らかの型をもつ要素に対応し、要素が属する集合は状態遷移規則の状態に対応すると考える。そのとき集合の値を変化させるスキーマはイベントに対応する。

イベントは複数の状態遷移を起こし、状態遷移はそれぞれ遷移条件をもつ。状態遷移が起こせない場合はエラー処理が行われる。遷移条件はイベントに対応するスキーマを分離し、各状態遷移に対応するスキーマ (遷移スキーマ) を求め、遷移条件と制約式が一致するか検証する。また遷移スキーマの規定式からエラー処理を抽出し、状態遷移規則のエラー処理と一致するか検証する。

以下、検証の具体的な手順を示す。

3.1 状態スキーマの宣言

検証者は仕様における状態スキーマがどのスキーマに当たるかを宣言する。複数の状態スキーマがある場合は合成したものを状態スキーマとする。

3.2 状態と集合の対応付け

検証者は状態遷移規則の各状態に対応する仕様中の集合を入力する。各エンティティの状態に対応する集合が

†電気通信大学大学院 電気通信学研究科 情報通信工学専攻
‡電気通信大学 電気通信学部 情報通信工学科

互いに素であり、ある種類のエンティティに対応づけられた集合全ての和集合が型となることを検証する。

もし互いに素でない集合があるか、和集合が型と一致しない場合、対応付けに誤りがあるか、どちらかの仕様で状態や集合の欠落が存在すると判定し検証者に報告する。また複数の変数をもつ型に対応する種類の状態遷移規則がない場合、状態遷移規則が必要な可能性があると判断し、検証者に確認する。

3.3 スキーマの適用による要素の移動の検出

仕様中の全ての操作スキーマについて、スキーマの適用によって起こりうる全ての要素の移動を検出する。ある操作スキーマにおいて式1が真であれば要素の移動が起こりうると判定する。X は検査対象の型、S_{x1} は遷移元の集合、S'_{x2} は遷移先の集合を示す。

$$\exists x : X \bullet x \in S_{x1} \land x \in S'_{x2} \quad (1)$$

3.4 イベントとスキーマの対応付け

イベントの起こす状態遷移に対応する要素の移動を過不足なく起こすスキーマをそのイベントに対応するイベントスキーマとする。

対応するスキーマが存在しないイベントがある場合は、スキーマの欠落や述語の誤りもしくは余分なイベントであると判定し、そのイベントと対応するスキーマの候補を検証者に提示する。一つのイベントに対応するスキーマが複数ある場合は、全く異なるスキーマが偶然同じ遷移を起こしている可能性が考えられるため、検証者に提示し一つのイベントには一つのスキーマが対応するように訂正を行う。

イベントに対応しない操作スキーマが存在した場合、そのスキーマがなんらかのイベントスキーマの一部分でなければ状態遷移規則においてイベントの抜けが発生していると考えられる。そのため、対応するイベントのないスキーマを検証者に提示する。

3.5 遷移スキーマの計算

イベントスキーマを分割し、遷移スキーマを求める。

状態遷移を起こすエンティティ X に対応する集合の要素を x 、あるイベントを E とし、E に対応するイベントスキーマを ES とする。E によって引き起こされる状態遷移の任意の一つを T とする。T の遷移元状態と遷移先状態にそれぞれ対応する集合を S_{T1}, S_{T2} とする。

この時、E によって X が状態遷移 T を起こすことは ES によって x が S_{T1} から S_{T2} へ移動することと対応する。よって ES によって S_{T1} から S_{T2} へ x が移動するための条件を計算することによって T の遷移スキーマ TS_T を求められる。これを式2に示す。

$$TS_T \mathrel{\widehat{=}} ES \land x \in S_{T1} \land x \in S'_{T2} \quad (2)$$

すべての状態の組合わせで遷移スキーマを求め、遷移スキーマから制約式、規定式を求める。

3.6 遷移条件の比較

状態遷移の遷移条件と遷移スキーマの制約式を比較し、同意であるか検証を行う。遷移条件の記述はZ記法によるが、条件変数はZ仕様とは異なるものが使われており、そのままでは比較できない。そこで、検証者は条件変数に対してZ仕様における述語との対応付けを行う。

検査対象の遷移スキーマをTSとし、制約式をX、遷移条件をYとする。式3が真であれば、制約式と遷移条件は同意であると判定できる。

$$\forall TS \bullet X \Leftrightarrow Y \quad (3)$$

すべての遷移スキーマで真となれば誤りはないと判断する。遷移条件と制約式が同意でないと判断された場合は、どちらかの式に過不足があるためであり、検証者に相違点を提示する。

3.7 エラー処理の比較

エラー処理に関する誤りはエンティティの種類ごとにZ仕様から次のように検出する。

1. 対象エンティティの種類以外のエンティティはOKを返す状態に固定し、対象エンティティのそれぞれの状態において規定式からエラー処理に対応する式を算出する。
2. 1の結果からZ仕様によるエラー処理表を作成する。
3. 状態遷移規則のエラー処理表と比較を行い、同じ配列であるか検査する。

複数の状態で同じエラー処理が繰り返し使われるため、エラー処理の配列を見ることで一致を確認できる。

エラー処理表の作成では他種類のエンティティの状態を固定し計算していた。全てのエンティティの種類に対して一致を確認できると、次に検査を行っていない状態の組み合わせでエラー処理の検出を行う。検出したエラー処理が組み合わせた状態のいずれかに対応するエラー処理であれば、全てのエラー処理は正しいと判定する。

エラー処理表が一致しない場合、双方のエラー処理表からエラー処理の対応付けを表示し、対応付けが正しいか検証者が確認し、誤った対応付けがされていた場合対応付けを訂正する。正しい対応付けがされていれば一致しない箇所でZ仕様か状態遷移規則に誤りがあると判断し、その箇所を検証者に提示する。

4. 考察

本手法ではZ仕様と状態遷移規則の比較により誤りを検出する。Z仕様と状態遷移規則は性質が異なるため、同じ誤りが含まれる可能性は少ないと考えられるが、同じ誤りが含まれていた場合は検出することができない。

検出された誤りは相違点から誤った可能性のある箇所は特定できるが、どちらの仕様の誤りによるものかはわからないため、訂正の支援を行うことは難しいと考えられる。また、検証において検証者の入力量が多いため、候補の表示を行うなど支援を行う必要がある。

5. おわりに

本研究ではZ仕様と状態遷移規則の比較により相違点を見つけ、誤りを検出する手法を提案した。今後は具体的な仕様を用いた本手法の評価を行っていく予定である。

参考文献

[1] 小池 孝政, 織田 健. システム構成要素の状態遷移規則に着目した仕様の妥当性検証法. 日本ソフトウェア科学会第17回大会論文集, 2000.

[2] 齋藤 修一, 織田 健. 状態遷移規則を知識とした形式的仕様における制約条件の欠落検出法. 日本ソフトウェア科学会第18回大会論文集, 2001.