委員会特別寄稿

コードからSTMへ リバースRExSTM for C ツールについて

名古屋大学 大学院情報科学研究科 **吉田 則裕**
JASA状態遷移設計研究WG 主査 **青木 奈央**

1. はじめに

近年、組み込みソフトウエア開発の現場では、機能の多様化や高品質の要望に伴いレガシーコードの複雑化・肥大化やドキュメントの陳腐化に起因する問題が多く起こっている。例えば、設計資料がソースコードのみであったり、担当者が不在で既存のコードを修正すると派生して他の問題が発生してしまう場合である。

JASAの状態遷移設計研究WGでは、この問題を解消すべく、以前よりレガシーコードから状態遷移表をリバースモデリングする手法の確立に以前より取り組んでおり、その一環としてリバースエンジニアリングを補助するツールRExSTM for C を開発している。本稿では、状態遷移設計研究WGの取組みとREx STMツールについて紹介する。

2. 状態遷移設計研究WGの取組み

状態遷移設計研究WGでは、ソフトウエア開発の効率化のための手法導入が進まないことに危惧しており、レガシーコードのブラックボックス化が進んでいることに着目してきた。そこで、レガシーコードを解析し仕様を明確にすることでこの問題が解決できるのではと考えた。研究テーマを考えるうえで、重要になってくるのは仮説である。我々は、仮説として、「状態遷移設計は普遍的なモデルである。」と「レガシーコードの中に状態遷移は必ずある。」ということを打立て、状態がある場所の検討を始めた。検討後に至ったのは、「フラグのあるところに状態がある。」ということである。今回、我々は、レガシーコードから状態遷移表をリバースモデリングする「状態遷移表のリバースモデリングへの適応」を研究テーマと決定しREx STM for Cを開発した。

図1. 入力ソースコードの例

```
void task(void){                              //f1
  int t = in1, s = in2;                       //p1,p2
  if(t == 1){                                 //c1
    S++;                                      //p3
    if(state = S1){                           //c3
      out = S;                                //p4
    }else if(state == S2){      //c4
      out = 0;                                //p5
      state = S3;                             //p6
    }else{                                    //c5
          if(out == 0)                        //c6
        state = S1;                           //p7
    }
  }
}
```

3. 状態遷移表抽出手法

状態遷移表作成の流れを図1～4に示す。以下、提案手法の流れを示す。

手順1. ソースコードから条件処理表を作成
手順2. 条件処理表から状態変数を選択
手順3. 選択した状態変数に関する状態遷移表を作成

3.1 条件処理表

条件処理表とは、ある条件において実行される処理を関数ごとにまとめた表である。

ソースコードのある条件分岐文とそれに対応する処理を、条件分岐文の階層構造にしたがって表現した条件と処理の対応表である。

図2. 条件処理表の例

条件			処理
無条件実行			t = in1; s = in2;
t==1	無条件実行		S++;
	state==S1		out = S;
	state==S2		out = 0; state = S3;
	else	out==0	state = S1;

状態遷移表を抽出するための中間状態として条件処理表を作成する。条件処理表の作成例を図2に示す。条件処理表は、図2のように条件部と処理部に分割される。

条件部は、ソースコード内の関数および条件分岐文(if、elseif、elseおよびswitch)から作成される。条件部について、ソースコードから条件処理表に変換するための具体的な変換手順を説明する。まず、ifおよびelseifについては、その条件式を条件部に記述する。elseの場合は、表にELSEと記述することとする。switchについては、switch(var)で指定された変数varと、case num:のようにラベルで指定された値numを用いて、表にvar == numのように記述する。ラベルがdefaultの場合については、ELSEと記述する。処理部は、ある条件部に対応する処理を記述する。

3.2 状態変数の選択

条件処理表を作成した後、条件処理表から状態変数を見つけ、全ての状態変数の中から、状態遷移表で表現したい状態変数を選択する。本手法では、状態変数を1つだけ選択する場合を考える。

ここで、状態変数の定義は、以下のとおりである。

- ●状態変数は分岐条件に使用されている。
- ●状態変数が取りうる値は有限個である。
- ●状態変数は内部で更新される。

たとえば、図2の条件処理表における状態変数候補は、変数stateである。変数stateは、条件式に含まれており、かつそのスコープ内で値が更新されているため、stateを状態変数として選択することができる。

ただし、状態変数は必ず1つの変数になるとは限らない。if(a == 1 && b == 0)のような条件分岐文では、その条件分岐文のスコープの内で、aかb、またはその両方の値が更新されているなら、a && bを1つの状態変数として扱う。

3.3 状態遷移表の作成

選択された状態変数について、状態遷移表を作成する。状態遷移表の作成は、条件処理表をもととして行われる。

まず、条件処理表の処理部の変更について説明する。処理部において、状態変数が取りうる値を条件部から抽出し、その状態の数だけ条件処理表の処理部の列を複製する(図3)。

変数stateは、S1、S2およびELSEの3つの値

図3. 条件処理表の例

条件	state		
	S1	S2	ELSE
無条件実行	t = in1; s = in2;	t = in1; s = in2;	t = in1; s = in2;
t==1	S++;	S++;	S++;
	out = S;	out = S;	out = S;
	out = 0; state = S3;	out = 0; state = S3;	out = 0; state = S3;
	state = S1;	state = S1;	if(out == 0) state = S1;

を取りうるため、同じ処理部を3列作成し、各列を、stateがS1、S2およびELSEのそれぞれの値の場合に実行されうる処理に対応させる。

すると、図3においては、条件列には状態と重複する条件式があり、それに対応する処理も不要なものが存在する。このように、不要な箇所が図3中において、黒く塗り潰されている。不必要な箇所を削除し、表を整形すると、図4のような状態遷移表を得ることができる。

4. RExSTM for C

RExSTM for Cとは、レガシーコードから状態遷移表を自動で生成するツールである。

A)C言語のソースコードの再フォーマット、および構文解析。

B)構文解析結果から条件処理表を作成条件処理表から状態変数を選択するためのUIの実装

C)選択された状態変数に関する状態遷移表の作成。

3節において説明した手順と比べ、C言語のソースコードを一定の形式に整形する作業が追加されている。

4.1 構文解析と条件処理表の作成

解析対象のソースコードの整形およびその構文解析の方法を述べる。以下にその大まかな流れを示す。

①GCCによってマクロ展開し、コメントを削除する。
②プログラムの形式の整形を行う。
③条件分岐文に対する識別子の付与を行う。
④条件式を``無条件"としたif文の追加を行う。
⑤グローバル変数・関数プロトタイプの宣言、#pragmaおよびインラインアセンブラの削除を行う。
⑥状態変数候補を抽出する。
⑦条件分岐文の構造解析を行う。
⑧TSV形式の条件処理表を出力する。
⑨Excel VBAでTSV形式の条件処理表を読み込み、表を作成する。

図4. 状態遷移表の例

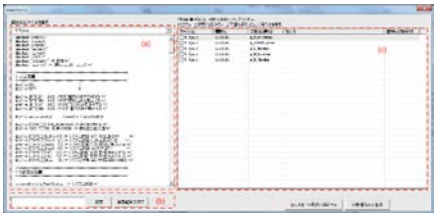
イベント	state		
	S1	S2	ELSE
無条件実行	t = in1; s = in2;	t = in1; s = in2;	t = in1; s = in2;
t==1	S++;	S++;	S++;
	out = S;	out = 0; state = S3;	if(out == 0) state = S1;

4.2 状態変数の選択

条件処理表を作成した後は、状態変数候補の中から状態変数を選択し、状態遷移表を出力する。このとき、状態変数の選択をツールが行わず、ユーザが行うものとする。その理由は、あるソースコードには複数の状態変数候補が存在し、どの状態変数を選択すれば有効な条件処理表を作成できるかの判定を、ユーザとの対話型インタフェースによって実装するためである。

そこで、Excel VBAによって、状態変数候補を選択するための補助を目的とした、フォームアプリケーションを実装する。フォームアプリケーションの概観を図5に示す。

図5. 状態変数候補選択画面



以下、フォームアプリケーションの機能を説明する。

(a)ソースコードプレビュー機能
入力されたソースコードを表示する。

(b)状態変数検索機能
状態変数名を入力し、検索することができる。

(c)状態変数選択機能
状態変数候補の中から、状態変数として選択する変数を選択することができる。状態変数に対してコメントを付与する機能も搭載している。

4.3 状態遷移表の出力

選択された状態変数についての状態遷移表を出力する。まず、条件処理表から不要な条件分岐文を削除する。次に、選択した状態変数が取りうる値それぞれ に対して、実行される処理文と実行されない処理文を抽出する。実行されない処理は不要であるため、黒く塗りつぶす。最後に、表を整理して状態遷移表が完成する。

5. 考察

このツールの適用効果として4つがあげられる。1. 機械的にソースコードから状態遷移表を作成するプロセスがみえてきた。2. 振舞いの可視化、モデル化によりレビューがしやすくなった。振舞いの漏れ抜けが発見できるようになった。3. 状態変数名の変更など、リファクタリングの要素が抽出可能になった。4. 言語に依存せず、すべてのコードに適用可能であることがわかった。今回のツールはC言語対応としているが、他のJAVAなどの言語でも十分適用可能であると考える。

現状のツールで状態遷移表が生成できることにより、第三者でも容易に仕様が理解できるようになる。また、状態遷移表を主要に扱うツール(ZIPC)などと連携をはかることで、仕様の変更だけでなく高品質なコード、製品を顧客へ提供可能になり、より有効に活用できると考える。

6. 今後の展開

RExSTM for Cでは、仮説の裏付け、手順が適用できることは証明されたが、様々なパターンに適用できるかの検証は不十分である。実際に変換が困難なパターンとして、「状態の階層化パターン」if-elseで条件が階層化されている場合、状態遷移表も階層構造となる場合、「状態変数の演算パターン」andやorで演算してから判定している場合、演算式を状態変数として判定する場合、「状態変数の置換えパターン」A&BをCに代入して、Cで判定する場合に、変数の変更はA、Bで行っている場合、「状態変数の関数渡しパターン」A&BをCに代入して、Cで判定しているとき、変数の変更は関数内で行っている場合、「状態の並列化パターン」状態が並列である場合の優先順位、「イベント条件と内部条件」状態変数以下の層の分岐条件をイベントでなく処理する場合が考えられる。今後は、自動化できる部分はツール内で実行できるように進める。例えば、ソースから条件・処理対応表を自動生成や状態変数候補自動抽出、選択後、状態遷移表の自動生成をめざす。

吉田則裕:yoshida@ertl.jp
2004年九州工業大学情報工学部知能情報工学科卒業。2009年大阪大学大学院情報科学研究科博士後期課程修了。博士(情報科学)。現在、名古屋大学大学院情報科学研究科准教授。

青木奈央:nao@zipc.com
1998年BS in Mathematics、American University、Washington D.C. IPA研究員、北陸先端科学技術大学院大学 研究員などを経て現在、キャッツ株式会社プロダクト事業部に所属

下記のURL(アンケート)にお答えいただけますと、ツールをダウンロードすることが可能になります。
アクセスできない場合は、designwg@jasa.or.jpにメールを送付ください。
https://goo.gl/forms/AZGgqdkYfTA9nwSu1