シンボリック実行に基づく状態遷移表の抽出手法の提案

山本 椋太[1,a)]　清水 貴裕[1]　吉田 則裕[1]

概要：我々は，従前よりレガシー化した組込みソフトウェアの理解を支援するための状態遷移表の抽出を試みている．しかし，サンプルソースコードが少ないことにより，状態遷移表を得るための制御フローの導出規則が不足している問題がある．本稿では，シンボリック実行によって状態遷移表におけるイベントおよび状態の導出を行い，状態遷移表を作成する方法について検討を行う．

1. はじめに

我々は，組込みシステム開発の現場におけるソフトウェアの再利用性や保守性の向上の要求 [1] を達成するべく，レガシーコードに注目している．そこで，レガシーコードからリバースエンジニアリングによって状態遷移表を抽出することで，その理解を支援する手法を検討してきた [2]．しかし，状態遷移表抽出のための制御フロー導出規則を検討するためのサンプルソースコードが不足している．また，状態値をソースコードにおける条件式から推定することが困難であり，等値以外の条件式における状態遷移表の状態値の表記を決定することが難しいという問題がある．

ところで，シンボリック実行では，ある変数を具体値を伴わない “シンボル” として扱い，疑似的にプログラムを実行する．そして，各実行パスを通る制約を求め，条件分岐に影響するシンボルの制約（パス条件）を導出する．導出されたすべてのパス条件を解くことで，パスを網羅したテストケースを生成することができる [3]．加えて，シンボリック実行によって，確実に満たされない制約式によるパスを削除しつつ，パスを抽出することが可能となるため，実行されうるパスだけに注目して出力することができる．

本稿では状態遷移表を抽出可能なソースコードを増やすべく，シンボリック実行ツールを利用して状態遷移表の抽出を行うことを検討する．

本稿においては，LLVM コンパイラ環境を用いたシンボリック仮想機械である KLEE[4][*1] を使用する．KLEE の解析においては，中間言語として LLVM を使用しているため，多言語対応が容易であるという利点があるが，ポインタや浮動小数点演算においての制約がある [3]．

KLEE では，シンボリック実行によって最終的に求められた制約式から，テストのための具体値を抽出し，テスト支援を行うことも可能である．この具体値も利用していくことを考えている．

2. 提案手法

KLEE を用いて状態遷移表を作成するための手法について説明する．手法適用の流れを図 1 に示す．図 1 中で，“$Sym\_$【変数名】” となっているものは，指定された変数名に対するシンボルである．

まず，全ての変数をシンボルに指定して，すべての実行パスにおける制約式を KLEE によって得る．同時に，ある実行パスにおける処理も全て抽出し，制約式と処理を対応付ける．これによって，条件シーケンス表を作成する．条件シーケンス表は，表の左側に制約式，右側に制約式に対応する処理をまとめた表である．

次いで，状態変数を 1 つ指定する．状態変数とは，システムの状態を表すような値を取る変数であり，先行研究 [2] と同様，解析者が指定する．解析者は状態変数を考えるための材料として，条件シーケンス表を利用する．

そして，すべての制約式から状態変数を含むものを抽出し，状態遷移表における状態列として出力するが，条件式が複雑だと状態値が煩雑になる．そこで，KLEE のテストケースの出力機能を活かし，代表値を併記することで可読性を向上する．また，抽出元である制約式の集合からは抽出した式を削除し，イベント行として出力する．加えて，状態変数として選択した変数を含む処理文を抽出し，表記上の差異を与える．図 1 における状態遷移を表している表記上の差異は，“<<< >>>” によって囲っている箇所である．このようにして，通常の処理と遷移を表記上区別する．遷移を含まないようなイベントと状態の組み合わせが存在する場合，そのとき自状態に遷移するか，終了するかを，そ

[1] 名古屋大学
[a)] muku@ertl.jp
[*1] https://klee.github.io/

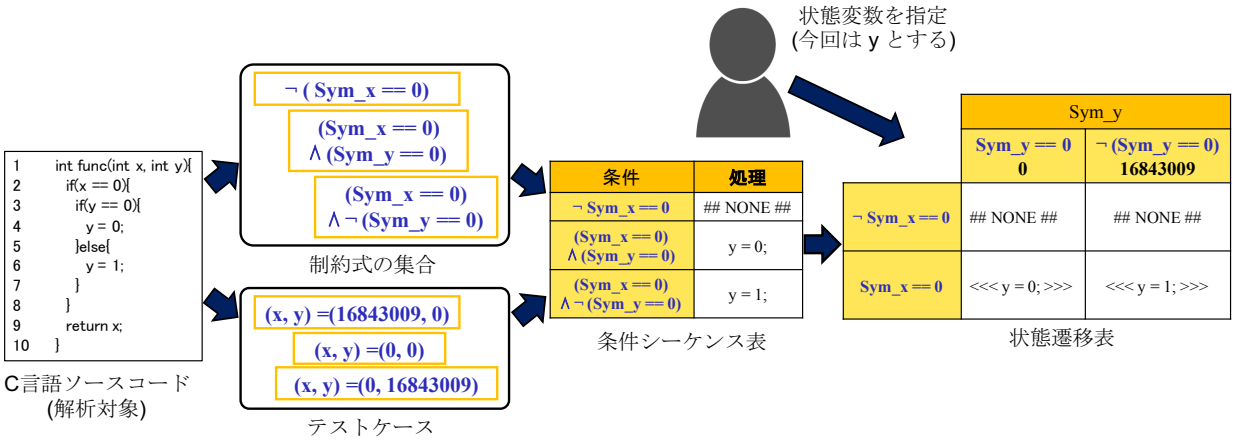

図 1 提案手法の適用手順

のパスがループするのかどうかから決定する．すなわち，ある実行パスを経たとき，return や break するか，または単純にループから抜けるのかということを確認して遷移先を決定する．条件シーケンス表に対し上記の変更を行うことで状態遷移表を抽出可能だと考えている．

本手法では，対象ソースコードに制約を設ける．まず，シンボリック実行上の制約がある．シンボリック実行する上では，ループが実行時間に影響を与える．そのため，ループ回数およびループ内の処理がいずれかであっても多い場合，解析に時間を要する．また，割込みに関しても実行時間上の制約がある．割込みは，実行パスとしての表現は可能だが，割込み禁止でない箇所すべてにおいて割込みが生じる可能性があるとするなら，解析パスの規模が大きくなるため実行時間上の問題が生じる．

次に，KLEE の制約がある．KLEE は，浮動小数点数型の値をすべて 0 と判定する．また，ポインタについてもアドレス空間をすべて探索する可能性について制約がある．そのため，浮動小数型およびポインタ型の変数が条件分岐文の条件に影響しておらず，シンボルにする必要がないことが制約となる．

また，本提案手法は従来の研究 [2] における問題点の改善につながる．従来の研究では，イベントや状態の判定における ELSE の範囲を抽出することが困難である．完全に同一の if-else if-else のブロックであればまとめることは可能であるが，そうでなければ，まとめることができず，冗長な状態遷移表となる．そのため，本提案手法は，従来の研究の状態遷移表を改善することにつなげられる．

3. おわりに

提案手法によって，従来手法では対応しきれていない比較演算子を含む条件式にも対応することが可能であると考えている．従来手法では，比較演算子の内，等値かつ左辺か右辺のどちらかが定数であり演算子を含まないなど厳しい制約が存在するが，シンボリック実行によってこれらが解決すると考えている．加えて，KLEE は LLVM に変換可能なプログラミング言語に対応することが可能であるため，多言語対応が可能であるという利点が存在する．

しかしながら，既述の通り KLEE における解析可能なソースコードの制約が存在する．

現在，本研究では，KLEE によって内部的に生成されている構文木の抽出方法を模索している．構文木が抽出できれば，既存研究の成果を活かしてツール化することが可能であると考えている．ただし，ツールの実装前に KLEE によるシンボリック実行時間の計測や，C 言語における様々な構文に対する解析結果を議論する必要がある．そして，可読性についても検討する必要がある．実行されないパスを考慮しないため，すべてのパスを網羅することに比べると表の規模は小さくなるが，イベントが複雑な数式になるなど，状態遷移表としての可読性が低下する可能性がある．

最後に，形式手法による検証にも応用可能だと考えている．KLEE を利用することで，従来手法による制約式の抽出よりも探索空間を狭めることが可能になると考えており，その結果，形式手法における状態爆発が生じる可能性を低減できると考えている．

参考文献

[1] 高田広章：組込みシステム開発技術の現状と展望，情報処理学会論文誌， Vol. 42, No. 4, pp. 930–938 (2001).
[2] 山本椋太，吉田則裕，青木奈央，高田広章：組込みソフトウェアを対象とした状態遷移表の抽出と分析支援の検討，電子情報通信学会技術研究報告，Vol. 117, No. 136, pp. 133–138 (2017).
[3] 徳本 晋，上原忠弘，宗像一樹，菊地英幸，江口 亨，石田晴幸，馬場匡史：C/C++ シンボリック実行ツール KLEE の適用と CPPUNIT 連携ツールの開発，組込みシステムシンポジウム 2011 論文集，Vol. 2011, pp. 23–1–23–8 (2011).
[4] Cadar, C., Dunbar, D. and Engler, D. R.: KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs., *Proc. of OSDI 2008*, Vol. 8, pp. 209–224 (2008).