■付録１　研究スケジュール

本研究は以下のスケジュールで行った.

項目	内容	期間
論文調査	CREST に関する論文の調査	2014 年 5 月
CREST のインストール	VMWare による Linux 環境の構築と, CREST 及び必要プログラムのインストール	2014 年 5 月～ 2014 年 6 月
CREST の試用	サンプルコードにて CREST を実行して, 出力結果を確認	2014 年 7 月～ 2014 年 8 月
SIG の準備・実施	ソフトウェア品質シンポジウム SIG：Let’s Concolic Testing の準備と実施	2014 年 8 月～ 2014 年 9 月
バグの抽出方法をマスターする	バグの種類と CREST での摘出可能性についての調査	2014 年 8 月～ 2014 年 11 月
新旧プログラムの結果比較方法をマスターする	デグレード防止のための比較方法についての調査	2014年11月～ 2014 年 12 月
実施手順の整理	バグの抽出方法及び新旧プログラムの比較についてのまとめ	2014年12年～ 2015 年 1 月

■付録２　CREST 検証環境

CREST 検証環境を以下に記す.

ホスト OS	Windows7(32bit), Windous8(32bit)
VM	VMware-6.0.2
ゲスト OS	Ubuntu-14.0.4
Concolic-testing ツール	CREST-0.1.1
Crest 走行に必要なツール	Yices、CIL、OCAML

CREST は Linux または Mac OS X 上で動作する. 本研究では, Windows 上の仮想環境(VMware)で Linux を動作させることにした.

■付録３　三角形形状判定プログラムのソースコード：オリジナル

3.2 で示した三角形形状判定プログラムの仕様に基づき作成したプログラム．

```
/*  Concolic-Testing 例題：三角形の形状判断  */

#include <stdio.h>
#include <assert.h>

int triangle(int iSide1, int iSide2, int iSide3);

int main(void){
            /* main は関数を呼び出して結果をプリント */
            int a,b,c,m;
             printf("input 3 sides value =>");
            scanf("%d %d %d",&a,&b,&c);
            m = triangle(a,b,c);
            printf("Shape of Triangle = %d¥n", m);
            return 0;
}

/*  形状を判定する */
int triangle(int iSide1, int iSide2, int iSide3){
            int msg;
   /* 入力データ誤りのチェック */
            if(iSide1 <= 0 || iSide2 <= 0 || iSide3 <= 0)          {
                        printf("Input Data is invalid.¥n");
                        msg = 10;
                        return msg;
            }
   /* 三角形であることのチェック*/
            if ((iSide1 + iSide2)<= iSide3 ||  (iSide2 + iSide3)<= iSide1 || (iSide1 + iSide3)<= iSide2)
            {
                        printf("Sides do not form a legal triangle.¥n");
                        msg = 10;
                        return msg;
            }
   /* 三角形の形状判定*/
            else{
                msg = 20; /* 不等辺三角形  */
                        if( iSide1 == iSide2 || iSide2 == iSide3 || iSide1 == iSide3){
                                    msg = 30;     /* 二等辺三角形    */
                                    if (iSide1 == iSide2 && iSide1 == iSide3){
                                                msg =40;     /* 正三角形 */
                                    }
                        }
                return msg;
            }
}
```

■付録 4　 CREST 対応した三角形形状判定プログラムの解析とテスト実行結果
(1) ソースコード
　ソースコード名：triangle.c

```c
/*  Concolic-Testing 三角形の形状判断  */

#include <crest.h>
#include <stdio.h>
#include <assert.h>
#define DEBUGMD

int triangle(int iSide1, int iSide2, int iSide3);

int main(void){
        /* main は関数を呼び出して結果をプリント */
        int a,b,c,m;
#ifndef DEBUGMD         /*  CREST 実行時はスキップ */
        printf("input 3 sides value =>");
        scanf("%d %d %d",&a,&b,&c);
#endif
        m = triangle(a,b,c);
        printf("Shape of Triangle = %d¥n", m);
        return 0;
}

/*  形状を判定する */
int triangle(int iSide1, int iSide2, int iSide3){
        int msg;
        CREST_int(iSide1);
        CREST_int(iSide2);
        CREST_int(iSide3);
  /* 入力データ誤りのチェック */
        if(iSide1 <= 0 || iSide2 <= 0 || iSide3 <= 0)        {
                printf("Input Data is invalid.¥n");
                msg = 10;
                return msg;
        }
  /* 三角形であることのチェック*/
        if ((iSide1 + iSide2)<= iSide3 ||  (iSide2 + iSide3)<= iSide1 || (iSide1 + iSide3)<= iSide2)
        {
                printf("Sides do not form a legal triangle.¥n");
                msg = 10;
                return msg;
        }
  /* 三角形の形状判定*/
        else{
           msg = 20; /* 不等辺三角形  */
                if( iSide1 == iSide2 || iSide2 == iSide3 || iSide1 == iSide3){
                        msg = 30;     /* 二等辺三角形    */
                        if (iSide1 == iSide2 && iSide1 == iSide3){
                                msg =40;     /* 正三角形 */
                        }
                }
           return msg;
        }
}
```

CREST 実行用に Symbolic 変数を宣言

CRSET 実行時に入力しなくて良いようにした

(2) run_crest で(4)の CREST 生成値を出力する方法

①CREST のプログラムにパッチを充てる　*本研究ではこちらを採用

・Crest の src/run_crest/concolic_search.cc にパッチを入れる.

Search::RunProgram () のところ.

LaunchProgram()処理の直前に次を入れる.

```
//Save the given inputs.
Char fname[32];
Snprintf(fname, 32 "input.%d", num_iters_);
WriteInputToFileOrDie(fname,inputs);
```

・crest を make する.

出力先：input ファイル

②印刷(printf)文を挿入する

・CREST 変数宣言の後ろに

printf("%d,%d,%d¥n",iSide1,iSide2,iSide3);

を挿入する.

```
CREST_int(iSide1);
CREST_int(iSide2);
CREST_int(iSide3);
```

出力先：(4)run_crest コマンドによるテスト実行

(3)CRESTC コマンドによるコード解析実行

```
mptqa@mptqa-virtual-machine:~/CREST/test/triangle$ /home/mptqa/CREST/crest-0.1.1/bin/crestc
triangle.c
gcc -D_GNUCC -E -I/home/mptqa/CREST/crest-0.1.1/bin/../i         -DCIL=1 triangle.c
-o ./triangle.i
/home/mptqa/CREST/crest-0.1.1/cil/obj/x86_LINUX/ci
--doCrestInstrument ./triangle.i
gcc -D_GNUCC -E -I/home/mptqa/CREST/crest-0.1.1/bin/../include ./triangle.cil.c -o ./triangle.cil.i
gcc -D_GNUCC -c -I/home/mptqa/CREST/crest-0.1.1/bin/../include -o ./triangle.o ./triangle.cil.i
triangle.c:11:1: warning: '__crest_skip__' attribute directive ignored [-Wattributes]
  /* main は関数を呼び出して結果をプリント */
・・・
・・・
・・・
gcc -D_GNUCC -o triangle -I/home/mptqa/CREST/crest-0.1.1/bin/../include ./triangle.o
/home/mptqa/CREST/crest-0.1.1/bin/../lib/libcrest.a -L/home/mptqa/CREST/crest-0.1.1/bin/../lib
-lstdc++
Read 22 branches.
Read 46 nodes.
Wrote 22 branch edges.
```

CRESTC のコマンド.
C プログラムを配置したディレクトリから，crestc を実行する.

このようにコメントなど CRSETC の解析では無視される warning がいくつか出るが，ここでは以降割愛する.

ブランチ数，ノード数，エッジ数が出力される.

(4)run_crest コマンドによるテスト実行

```
mptqa@mptqa-virtual-machine:~/CREST/test/triangle$ /home/mptqa/CREST/crest-0.1.1/bin/run_cr
est ./triangle 20 -dfs
Iteration 0 (0s): covered 0 branches [0 reach funs, 0 reach branches].
Input Data is invalid.
Shape of Triangle = 10
Iteration 1 (0s): covered 1 branches [1 reach funs, 22 reach bra
Input Data is invalid.
Shape of Triangle = 10
Iteration 2 (0s): covered 3 branches [1 reach funs, 22 reach branches].
Input Data is invalid.
Shape of Triangle = 10
Iteration 3 (0s): covered 5 branches [1 reach funs, 22 reach branches].
Shape of Triangle = 40
Iteration 4 (0s): covered 12 branches [1 reach funs, 2
Sides do not form a legal triangle.
Shape of Triangle = 10
Iteration 5 (0s): covered 13 branches [1 reach funs, 2
Sides do not form a legal triangle.
Shape of Triangle = 10
Iteration 6 (0s): covered 14 branches [1 reach funs, 22 reach branches].
Sides do not form a legal triangle.
Shape of Triangle = 10
Iteration 7 (0s): covered 15 branches [1 reach funs, 22 reach branches].
Shape of Triangle = 30
Iteration 8 (0s): covered 19 branches [1 reach funs, 22 reach branches].
Shape of Triangle = 30
Iteration 9 (0s): covered 20 branches [1 reach funs, 22 reach branches].
Shape of Triangle = 20
Iteration 10 (0s): covered 21 branches [1 reach funs, 22 reach bran
Shape of Triangle = 30
Iteration 11 (0s): covered 22 branches [1 reach funs, 22 reach branches].
```

run_crest のコマンド．
[20]は実行される Iteration 数以上を入力する必要がある．（このプログラムの場合，11 以上）
[-dfs]で深さ優先探索を指定．

各 Iteration はパスと入力値を組合せたテストの繰り返す数である．
Shape of Triangle = 10：三角形の形状を出力．
CREST で生成される辺長は通常表示されないので，別ファイルに出力されるようにした

CRESTC で算出された 22 個のブランチを 11 個の Iteration でカバーしている．

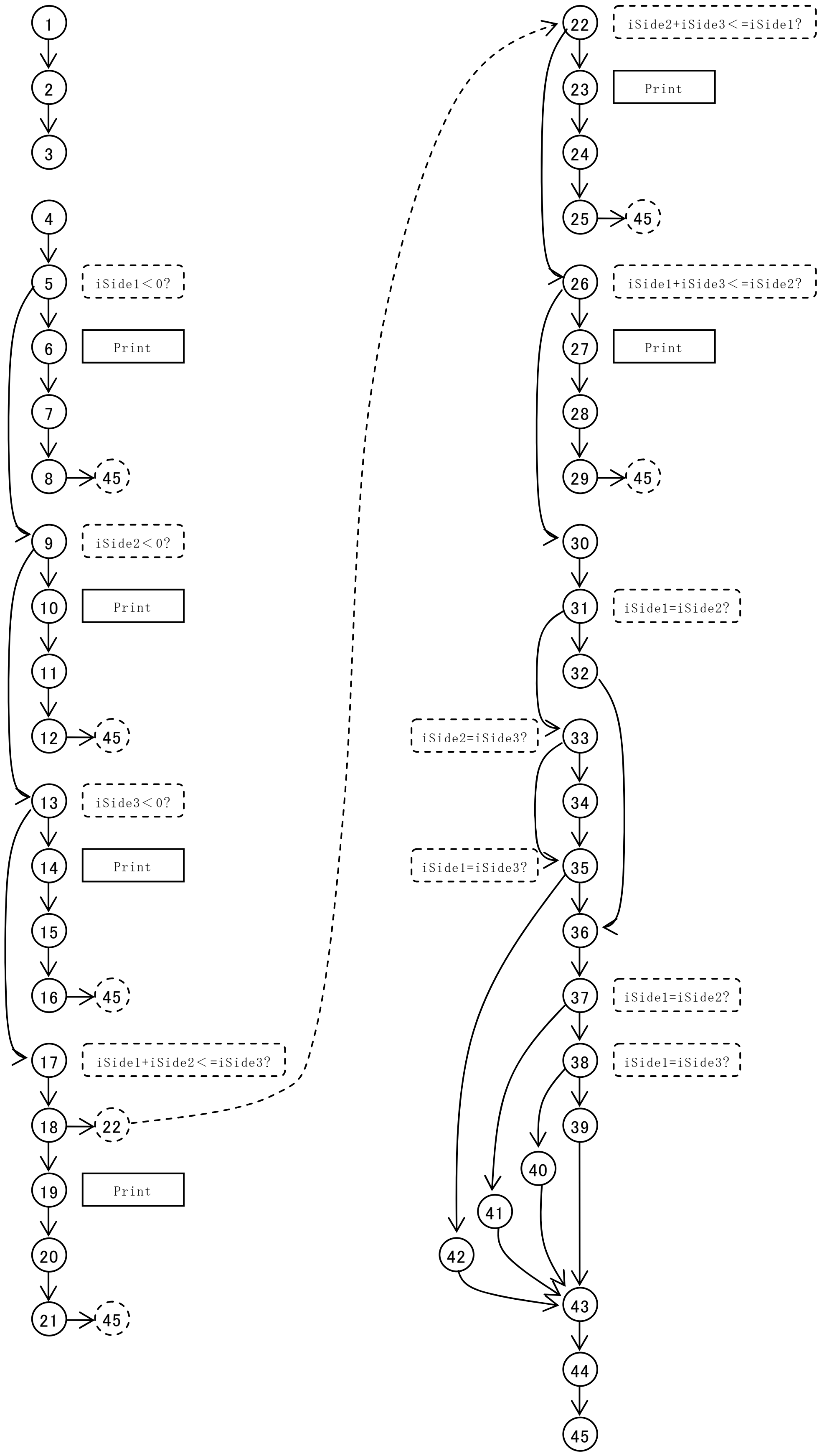
(5)run_crest コマンド実行時の CREST で生成された入力値
input1,・・・, input11 と別ファイルで出力されたものを整理する．
Iteration1 と input1,・・・, Iteration11 と input11 が対応している．

	input1	input2	input3	input4	input5	input6	input7	input8	input9	input10	input11
iSide1	0	1	1	1	1	2	1	2	3	4	2
iSide2	0	0	1	1	1	1	2	1	2	3	2
iSide3	0	0	0	1	2	1	1	2	2	2	1

(6)表 2.三角形形状判定プログラム:CREST 実行結果の DT との対応
run_crest コマンドによるテスト実行結果と run_crest コマンド実行時の CREST で生成された入力値を組み合わせる．

テストケース			1	・・・	11
原因	辺長	A	0	・・・	2
		B	0	・・・	2
		C	0	・・・	1
結果	Invalid(10)		X	・・・	
	三角形非形成(10)			・・・	
	正三角形(40)			・・・	
	二等辺三角形(30)			・・・	X
	不等辺三角形(20)			・・・	

(7)CRESTC コマンドで解析した中間言語展開後の制御パス
CREST の出力する Configuration（CFG）情報を使って作成する.



■付録 5　三角形形状判定プログラム(バグ有)の解析とテスト実行結果 1

(1)ソースコード

ソースコード名：triangle_bug.c

```
/*  Concolic-Testing 三角形の形状判断   */

#include <crest.h>
#include <stdio.h>
#include <assert.h>
#define DEBUGMD

...

   /* 三角形であることのチェック*/
        if ((iSide1 + iSide2)< iSide3 ||  (iSide2 + iSide3)< iSide1 || (iSide1 + iSide3)< iSide2)
        {
                printf(″Sides do n          legal triangle.¥n″);
                msg = 10;
                return msg;
        }
   /* 三角形の形状判定*/
        else{
            msg = 20; /* 不等
                if( iSide1 == iSide2 || iSide2 == iSide3 || iSide1 == iSide3){
                        msg = 30;     /* 二等辺三角形    */
                        if (iSide1 == iSide2 && iSide1 == iSide3){
                                msg =40;      /* 正三角形 */
                        }
                }
            return msg;
        }
}
```

if ((iSide1 + iSide2)<= iSide3 || (iSide2 + iSide3)<= iSide1 || (iSide1 + iSide3)<= iSide2)を
if ((iSide1 + iSide2)< iSide3 || (iSide2 + iSide3)< iSide1 || (iSide1 + iSide3)< iSide2)
とバグ有の状態にする.

(2)CRESTC コマンドによるコード解析結果

```
mptqa@mptqa-virtual-machine:~/CREST/test/triangle_bug$ /home/mptqa/CREST/crest-0.1.1/bin/cr
estc triangle_bug.c
gcc -D_GNUCC -E -I/home/mptqa/CREST/crest-0.1.1/bin/../include -DCIL=1 triangle_bug.c
-o ./triangle_bug.i
/home/mptqa/CREST/crest-0.1.1/cil/obj/x86_LINUX/cilly.asm.exe --out ./triangle_bug.cil.c
--doCrestInstrument ./triangle_bug.i
gcc -D_GNUCC -E -I/home/mptqa/CREST/crest-0.1.1/bin/../include ./triangle_bug.cil.c
-o ./triangle_bug.cil.i
gcc -D_GNUCC -c -I/home/mptqa/CREST/crest-0.1.1/bin/../include
-o ./triangle_bug.o ./triangle_bug.cil.i

...
...

gcc -D_GNUCC -o triangle_bug -I/home/mptqa/CREST/crest-0.1.1/bin/../include ./triangle_bug.o
/home/mptqa/CREST/crest-0.1.1/bin/../lib/libcrest.a -L/home/mptqa/CREST/crest-0.1.1/bin/../lib
-lstdc++
Read 22 branches.
Read 46 nodes.
Wrote 22 branch edges.
```

Warning は割愛

(3)run_crest コマンドによるテスト実行結果

```
mptqa@mptqa-virtual-machine:~/CREST/test/triangle_bug$ /home/mptqa/CREST/crest-0.1.1/bin/run_
crest ./triangle_bug 20 -dfs
Iteration 0 (0s): covered 0 branches [0 reach funs, 0 reach branches].
Input Data is invalid.
Shape of Triangle = 10
Iteration 1 (0s): covered 1 branches [1 reach funs, 22 reach branches].
Input Data is invalid.
Shape of Triangle = 10
Iteration 2 (0s): covered 3 branches [1 reach funs, 22 reach branches].
Input Data is invalid.
Shape of Triangle = 10
Iteration 3 (0s): covered 5 branches [1 reach funs, 22 reach branches].
Shape of Triangle = 40
Iteration 4 (1s): covered 12 branches [1 reach funs, 22 reach branches].
Sides do not form a legal triangle.
Shape of Triangle = 10
Iteration 5 (1s): covered 13 branches [1 reach funs, 22 reach branches].
Sides do not form a legal triangle.
Shape of Triangle = 10
Iteration 6 (1s): covered 14 branches [1 reach funs, 22 reach branches].
Sides do not form a legal triangle.
Shape of Triangle = 10
Iteration 7 (1s): covered 15 branches [1 reach funs, 22 reach branches].
Shape of Triangle = 30
Iteration 8 (1s): covered 18 branches [1 reach funs, 22 reach branches].
Shape of Triangle = 20
Iteration 9 (1s): covered 20 branches [1 reach funs, 22 reach branches].
Shape of Triangle = 30
Iteration 10 (1s): covered 21 branches [1 reach funs, 22 reach branches].
Shape of Triangle = 30
Iteration 11 (1s): covered 22 branches [1 reach funs, 22 reach branches].
```

(4)run_crest 実行時の各 Iteration の入力値

	input1	input2	input3	input4	input5	input6	input7	input8	input9	input10	input11
iSide1	0	1	1	1	1	3	1	2	3	1	2
iSide2	0	0	1	1	1	1	3	1	2	2	2
iSide3	0	0	0	1	3	1	1	1	1	1	1

■付録 6　三角形形状判定プログラム(バグ有)の解析とテスト実行結果 2

(1)ソースコード

ソースコード名：triangle_bug_add_assert.c

```
/*  Concolic-Testing 三角形の形状判断  */

・・・
・・・

  /* 三角形であることのチェック*/
        #ifdef     DEBUGMD
        assert( (iSide1 + iSide3)<= iSide2);
        #endif

        if ((iSide1 + iSide2)<= iSide3 ||  (iSide2 + iSide3)<= iSide1 || (iSide1 + iSide3)< iSide2)
        {
                printf("Sides do not form a legal triangle.¥n"
                msg = 10;
                return msg;
        }

・・・
```

assert 挿入.
ここでは単純にバグに対する条件文とした
が，バグの種類により assert を工夫する必要
がある.

(iSide1 + iSide3)<= iSide2)
を
(iSide1 + iSide3)< iSide2)
とバグ有の状態にする.

(2)CRESTC コマンドによるコード解析結果

```
mptqa@mptqa-virtual-machine:~/CREST/test/triangle_add+bug
_assert$ /home/mptqa/CREST/crest-0.1.1/bin/crestc triangle_add_bug_assert.c
gcc -D_GNUCC -E -I/home/mptqa/CREST/crest-0.1.1/bin/../include -DCIL=1 triangle_add_bug_assert.c
-o ./triangle_add_bug_assert.i
/home/mptqa/CREST/crest-0.1.1/cil/obj/x86_LINUX/cilly.asm.exe --out ./triangle_add_bug_assert.cil.c
--doCrestInstrument ./triangle_add_bug_assert.i
gcc -D_GNUCC -E -I/home/mptqa/CREST/crest-0.1.1/bin/../include ./triangle_add_bug_assert.cil.c
-o ./triangle_add_bug_assert.cil.i
gcc -D_GNUCC -c -I/home/mptqa/CREST/crest-0.1.1/bin/../include
-o ./triangle_add_bug_assert.o ./triangle_add_bug_assert.cil.i

・・・
・・・

gcc -D_GNUCC -o triangle_add_bug_assert
-I/home/mptqa/CREST/crest-0.1.1/bin/../include ./triangle_add_bug_assert.o
/home/mptqa/CREST/crest-0.1.1/bin/../lib/libcrest.a -L/home/mptqa/CREST/crest-0.1.1/bin/../lib
-lstdc++
Read 34 branches.
Read 66 nodes.
Wrote 38 branch edges.
```

Warning は割愛

(3)run_crest コマンドによるテスト実行結果

```
mptqa@mptqa-virtual-machine:~/CREST/test/triangle_add+bug
_assert$ /home/mptqa/CREST/crest-0.1.1/bin/run_crest ./triangle_add_bug_assert 20 -dfs
Iteration 0 (0s): covered 0 branches [0 reach funs, 0 reach branches].
Input Data is invalid.
Shape of Triangle = 10
Iteration 1 (0s): covered 1 branches [1 reach funs, 30 reach branches].
Input Data is invalid.
Shape of Triangle = 10
Iteration 2 (0s): covered 3 branches [1 reach funs, 30 reach branches].
Input Data is invalid.
Shape of Triangle = 10
Iteration 3 (0s): covered 5 branches [1 reach funs, 30 reach branches].
Shape of Triangle = 40
Iteration 4 (0s): covered 16 branches [1 reach funs, 30 reach branches].
Input Data is over.
Shape of Triangle = 5
Iteration 5 (0s): covered 17 branches [1 reach funs, 30 reach branches].
Input Data is over.
Shape of Triangle = 5
Iteration 6 (0s): covered 18 branches [1 reach funs, 30 reach branches].
Input Data is over.
Shape of Triangle = 5
Iteration 7 (0s): covered 19 branches [1 reach funs, 30 reach branches].
triangle_add_bug_assert: triangle_add_bug_assert.c:46: triangle: Assertion `(iSide1 + iSide3) >= iSide2'
failed.
Aborted (core dumped)
Iteration 8 (0s): covered 19 branches [1 reach funs, 30 reach branches].
Prediction failed!
Sides do not form a legal triangle.
Shape of Triangle = 10
Iteration 9 (0s): covered 20 branches [1 reach funs, 30 reach branches].
Sides do not form a legal triangle.
Shape of Triangle = 10
Iteration 10 (0s): covered 21 branches [1 reach funs, 30 reach branches].
Shape of Triangle = 30
Iteration 11 (0s): covered 25 branches [1 reach funs, 30 reach branches].
Shape of Triangle = 30
Iteration 12 (0s): covered 26 branches [1 reach funs, 30 reach branches].
Shape of Triangle = 20
Iteration 13 (0s): covered 27 branches [1 reach funs, 30 reach branches].
Shape of Triangle = 30
Iteration 14 (0s): covered 28 branches [1 reach funs, 30 reach branches].
```

assert によりバグ検知

(4)run_crest 実行時の各 Iteration の入力値

バグ有

	input1	input2	input3	input4	input5	input6	input7	input8	input9	input10	input11	Input12	Input13	Input14
iSide1	0	1	1	1	11	1	1	1	1	2	2	3	4	2
iSide2	0	0	1	1	1	11	1	3	1	1	1	2	3	2
iSide3	0	0	0	1	1	1	11	1	2	1	2	2	2	1

■付録７　割り算プログラムの解析とテスト実行結果２

(1) ソースコード

ソースコード名：divisin.c

```
/* 割り算プログラム */

#include <crest.h>
#include <stdio.h>
#include <assert.h>
#define DEBUGMD

int main(void){
        int a, b;
        CREST_int(a);
        CREST_int(b);

#ifndef DEBUGMD
        printf("input a b =>");
        scanf("%d %d", &a, &b);
#endif

        printf("input %d,%d¥n",a,b);
        if(a > 0){
                if(a < b){
                        printf("input a > b.¥n");
                        return 0;
                }else{
                #ifdef DEBUGMD
                        assert(b != 0);
                #endif
                        printf("quotient = %d¥n" , a / b);
                        printf("remainder = %d¥n" , a % b);
                }
        }
        return 0;
}
```

0 割検知用の assert
CREST の入力変数値自動生成により，0 となる条件が成立可能かを検証することが可能となる．

(2)CRESTC コマンドによるコード解析結果

```
mptqa@mptqa-virtual-machine:~/CREST/test/division$ /home/mptqa/CREST/crest-0.1.1/bin/cr
estc division.c
gcc -D_GNUCC -E -I/home/mptqa/CREST/crest-0.1.1/bin/../include -DCIL=1 division.c -o ./division.i
/home/mptqa/CREST/crest-0.1.1/cil/obj/x86_LINUX/cilly.asm.exe --out ./division.cil.c
--doCrestInstrument ./division.i
gcc -D_GNUCC -E -I/home/mptqa/CREST/crest-0.1.1/bin/../include ./division.cil.c -o ./division.cil.i
gcc -D_GNUCC -c -I/home/mptqa/CREST/crest-0.1.1/bin/../include -o ./division.o ./division.cil.i

・・・
・・・

gcc -D_GNUCC -o division -I/home/mptqa/CREST/crest-0.1.1/bin/../include ./division.o
/home/mptqa/CREST/crest-0.1.1/bin/../lib/libcrest.a -L/home/mptqa/CREST/crest-0.1.1/bin/../lib
-lstdc++
Read 6 branches.
Read 14 nodes.
Wrote 4 branch edges.
```

Warning は割愛

(3)run_crest コマンドによるテスト実行結果

```
mptqa@mptqa-virtual-machine:~/CREST/test/division$ /home/mptqa/CREST/crest-0.1.1/bin/run_cres
t ./division 20 -dfs
Iteration 0 (0s): covered 0 branches [0 reach funs, 0 reach branches].
input 0,0
Iteration 1 (0s): covered 1 branches [1 reach funs, 6 reach branches].
input 1,0
division: division.c:25: main: Assertion `b != 0' failed.
Aborted (core dumped)
Iteration 2 (0s): covered 1 branches [1 reach funs, 6 reach branches].
Prediction failed!
```

assert によりバグ検知

■付録８　仕様追加した三角形形状判定プログラムの解析とテスト実行結果

(1)ソースコード

ソースコード名：triangle_add.c

```
1   /*  Concolic-Testing 三角形の形状判断   */

    ...

26    /* 入力データ誤りのチェック */
27          if(iSide1 <= 0 || iSide2 <= 0 || iSide3 <= 0
28                  printf("Input Data is invalid.¥n"
29                  msg = 10;
30                  return msg;
31          }

    /* 上限チェック */
          if(iSide1 > 10 || iSide2 > 10 || iSide3 > 10)        {
                  printf("Input Data is over.¥n");
                  msg = 5;
                  return msg;
          }

34    /* 三角形であることのチェック*/
35          if ((iSide1 + iSide2)<= iSide3 ||  (iSide2 + iSide3)<= iSide1 || (iSide1 + iSide3)<= iSide2)
36          {
37                  printf("Sides do not form a legal triangle.¥n");
38                  msg = 10;
39                  return msg;
40          }

    ...
```

付録 4-26-31 行目の入力データ誤りチェックの後ろに
iSide1,2,3 が 10 より大きい時に 5 を出力する判定文を追加

(2)CRESTC コマンドによるコード解析実行

```
mptqa@mptqa-virtual-machine:~/CREST/test/triangle_add$ /home/mptqa/CREST/crest-0.1.1/bin/c
restc triangle_add.c
gcc -D_GNUCC -E -I/home/mptqa/CREST/crest-0.1.1/bin/../include -DCIL=1 triangle_add.c
-o ./triangle_add.i
/home/mptqa/CREST/crest-0.1.1/cil/obj/x86_LINUX/cilly.asm.exe --out ./triangle_add.cil.c
--doCrestInstrument ./triangle_add.i
gcc -D_GNUCC -E -I/home/mptqa/CREST/crest-0.1.1/bin/../include ./triangle_add.cil.c
-o ./triangle_add.cil.i
gcc -D_GNUCC -c -I/home/mptqa/CREST/crest-0.1.1/bin/../include
-o ./triangle_add.o ./triangle_add.cil.i
...
gcc -D_GNUCC -o triangle_add -I/home/mptqa/CREST/crest-0.1.1/bin/../include ./triangle_add.o
/home/mptqa/CREST/crest-0.1.1/bin/../lib/libcrest.a -L/home/mptqa/CREST/crest-0.1.1/bin/../lib
-lstdc++
Read 28 branches.
Read 59 nodes.
Wrote 28 branch edges.
```

仕様追加前の結果(付録 4)

```
Read 22 branches.
Read 46 nodes.
Wrote 22 branch edges.
```

仕様追加により，ブランチ数，ノード数，エッジ数が増加している．

(3)run_crest コマンドによるテスト実行

```
mptqa@mptqa-virtual-machine:~/CREST/test/triangle_add$  /home/mptqa/CREST/crest-0.1.1/bin/r
un_crest ./triangle_add 20 -dfs
Iteration 0 (0s): covered 0 branches [0 reach funs, 0 reach branches].
Input Data is invalid.
Shape of Triangle = 10
Iteration 1 (0s): covered 1 branches [1 reach funs, 28 reach branches].
Input Data is invalid.
Shape of Triangle = 10
Iteration 2 (0s): covered 3 branches [1 reach funs, 28 reach branches].
Input Data is invalid.
Shape of Triangle = 10
Iteration 3 (0s): covered 5 branches [1 reach funs, 28 reach branches].
Shape of Triangle = 40
Iteration 4 (0s): covered 15 branches [1 reach funs, 28 reach branches].
Input Data is over.
Shape of Triangle = 5
Iteration 5 (0s): covered 16 branches [1 reach funs, 28 reach branches].
Input Data is over.
Shape of Triangle = 5
Iteration 6 (0s): covered 17 branches [1 reach funs, 28 reach branches].
Input Data is over.
Shape of Triangle = 5
Iteration 7 (0s): covered 18 branches [1 reach funs, 28 reach branches].
Sides do not form a legal triangle.
Shape of Triangle = 10
Iteration 8 (0s): covered 19 branches [1 reach funs, 28 reach branches].
Sides do not form a legal triangle.
Shape of Triangle = 10
Iteration 9 (0s): covered 20 branches [1 reach funs, 28 reach branches].
Sides do not form a legal triangle.
Shape of Triangle = 10
Iteration 10 (0s): covered 21 branches [1 reach funs, 28 reach branches].
Shape of Triangle = 30
Iteration 11 (0s): covered 25 branches [1 reach funs, 28 reach branches].
Shape of Triangle = 30
Iteration 12 (0s): covered 26 branches [1 reach funs, 28 reach branches].
Shape of Triangle = 20
Iteration 13 (0s): covered 27 branches [1 reach funs, 28 reach branches].
Shape of Triangle = 30
Iteration 14 (0s): covered 28 branches [1 reach funs, 28 reach branches].
```

Iteration が 11 から 14 に増加.

(4)run_crest コマンド出力結果実行時の CREST で生成された入力値

	input1	input2	input3	input4	input5	input6	input7	input8	input9	input10	input11	Input12	Input13	Input14
iSide1	0	1	1	1	11	1	1	1	2	1	2	3	4	2
iSide2	0	0	1	1	1	11	1	1	1	2	1	2	3	2
iSide3	0	0	0	1	1	1	11	2	1	1	2	2	2	1

■付録 9　仕様追加した三角形形状判定プログラム(バグ有)の解析とテスト実行結果
(1)ソースコード
ソースコード名：triangle_add_bug.c

```
/*  Concolic-Testing 三角形の形状判断   */

...

  /* 入力データ誤りのチェック */
        if(iSide1 <= 0 || iSide2 <= 0 || iSide3 <= 0)         {
                printf("Input Data is invalid.¥n");
                msg = 10;
                return msg;
        }

 /* 上限チェック */
        if(iSide1 > 10 || iSide2 > 10 || iSide3 > 10)         {
                printf("Input Data is over.¥n");
                msg = 5;
                return msg;
        }

  /* 三角形であることのチェック*/
        if ((iSide1 + iSide2)<= iSide3 ||  (iSide2 + iSide3)<= iSide1 || (iSide1 + iSide3)< iSide2)
        {
                printf("Sides do not form a legal triangle.¥n");
                msg = 10;
                return msg;
        }
...
```

(iSide1 + iSide3)<= iSide2)
を
(iSide1 + iSide3)< iSide2)
とバグ有の状態にする.

(2)CRESTC コマンドによるコード解析結果

```
mptqa@mptqa-virtual-machine:~/CREST/test/triangle_add+bug$ /home/mptqa/CREST/crest-0.1.1/bin
/crestc triangle_add_bug.c
gcc -D_GNUCC -E -I/home/mptqa/CREST/crest-0.1.1/bin/../include -DCIL=1 triangle_add_bug.c
-o ./triangle_add_bug.i
/home/mptqa/CREST/crest-0.1.1/cil/obj/x86_LINUX/cilly.asm.exe --out ./triangle_add_bug.cil.c
--doCrestInstrument ./triangle_add_bug.i
gcc -D_GNUCC -E -I/home/mptqa/CREST/crest-0.1.1/bin/../include ./triangle_add_bug.cil.c
-o ./triangle_add_bug.cil.i
gcc -D_GNUCC -c -I/home/mptqa/CREST/crest-0.1.1/bin/../include
-o ./triangle_add_bug.o ./triangle_add_bug.cil.i

...
...

gcc -D_GNUCC -o triangle_add_bug
-I/home/mptqa/CREST/crest-0.1.1/bin/../include ./triangle_add_bug.o
/home/mptqa/CREST/crest-0.1.1/bin/../lib/libcrest.a -L/home/mptqa/CREST/crest-0.1.1/bin/../lib
-lstdc++
Read 28 branches.
Read 59 nodes.
Wrote 28 branch edges.
```

Warning は割愛

(3)run_crest コマンドによるテスト実行結果

```
mptqa@mptqa-virtual-machine:~/CREST/test/triangle_add+bug$  /home/mptqa/CREST/crest-0.1.1/bin
/run_crest ./triangle_add_bug 20 -dfs
Iteration 0 (0s): covered 0 branches [0 reach funs, 0 reach branches].
Input Data is invalid.
Shape of Triangle = 10
Iteration 1 (0s): covered 1 branches [1 reach funs, 28 reach branches].
Input Data is invalid.
Shape of Triangle = 10
Iteration 2 (0s): covered 3 branches [1 reach funs, 28 reach branches].
Input Data is invalid.
Shape of Triangle = 10
Iteration 3 (0s): covered 5 branches [1 reach funs, 28 reach branches].
Shape of Triangle = 40
Iteration 4 (0s): covered 15 branches [1 reach funs, 28 reach branches].
Input Data is over.
Shape of Triangle = 5
Iteration 5 (0s): covered 16 branches [1 reach funs, 28 reach branches].
Input Data is over.
Shape of Triangle = 5
Iteration 6 (0s): covered 17 branches [1 reach funs, 28 reach branches].
Input Data is over.
Shape of Triangle = 5
Iteration 7 (0s): covered 18 branches [1 reach funs, 28 reach branches].
Sides do not form a legal triangle.
Shape of Triangle = 10
Iteration 8 (0s): covered 19 branches [1 reach funs, 28 reach branches].
Sides do not form a legal triangle.
Shape of Triangle = 10
Iteration 9 (0s): covered 20 branches [1 reach funs, 28 reach branches].
Sides do not form a legal triangle.
Shape of Triangle = 10
Iteration 10 (0s): covered 21 branches [1 reach funs, 28 reach branches].
Shape of Triangle = 30
Iteration 11 (0s): covered 25 branches [1 reach funs, 28 reach branches].
Shape of Triangle = 30
Iteration 12 (0s): covered 26 branches [1 reach funs, 28 reach branches].
Shape of Triangle = 20
Iteration 13 (0s): covered 27 branches [1 reach funs, 28 reach branches].
Shape of Triangle = 30
Iteration 14 (0s): covered 28 branches [1 reach funs, 28 reach branches].
```

付録 8 と同様

(4)run_crest 実行時の各 Iteration の入力値

	input1	input2	input3	input4	input5	input6	input7	input8	input9	input10	input11	Input12	Input13	Input14
iSide1	0	1	1	1	11	1	1	1	2	1	2	3	4	2
iSide2	0	0	1	1	1	11	1	1	1	3	1	2	3	2
iSide3	0	0	0	1	1	1	11	2	1	1	2	2	2	1

入力値が仕様変更前の付録 4(5)input7 と違う．