TEHARI Ayoub
L3 informatique

le 22/03/2020

MOCA : utilisation de Klee (la suite)

Rendu pour la séance 8

modication du script env.sh :
Explication :
+la variable d'environnement PATH n'est pas mise à jour(ex : la commande klee est introuvable)
+la librairie kleeRuntest se trouve dans le chemin $KLEE/lib, celui-ci devra être ajouté à la variable LD_LIBRARY_PATH, pour que l'exécution marche d'un programme utilisant des fonctions comme klee_make_symbolic.
j'ai donc avant tout récupérer le fichier ~monniaud/env.sh sur le répertoire du projet et puis j'ai effectué la modification suivante :

```
PROF=/home/m/monniaud
PATH=$PROF/bin:$PATH
KLEE=$PROF/packages/klee/2017-09-26_c7a1f9d
LD_LIBRARY_PATH=$PROF/lib:$KLEE/lib:$LD_LIBRARY_PATH
export PATH
export LD_LIBRARY_PATH
export KLEE

LLVM_COMPILER=clang
export LLVM_COMPILER

PYTHONPATH=$PROF/.local/lib/python2.7/site-packages
export PYTHONPATH
```

L'utilisation de la commande 'source' reste la même (à chaque ouverture d'un nouveau shell)

Klee permet de générér des entrées pertinantes et puis les stocker dans des fichiers.ktest
parfois en cours de l'exécution symbolique, on rencontre des erreurs d'éxécution, klee dans ce cas va genérer en plus du fichier d'entrée, un autre fichier .err qui contient l'arbre des appels récursives concernant l'erreur rencontré.
ktest-tool permet d'interpréter un fichier (.ktest), contenant une entrée généré lors de l'exécution de klee
l'option –write-ints : découpe la variable symbolisé en blocs de 4 octets chacun et renvoie le contenu de chacun comme integer.

Retour au projet :
pour symboliser une chaîne de caractère de taille N, il faut faire appel à klee_make_symbolic, ensuite on a le choix entre :
soit on force le contenu de la dernière case à 0 ;

soit on rajoute « juste après » une contrainte du genre tab[N-1]==’\0’, par le biais de la fonction klee_assume

SINON :
on arrive à des erreurs de pointage, puisque l’application attend une entrée « valide », c’est-à-dire une chaîne de caractère qui se termine par le caractère de fin de chaîne.
Comme expliqué sur ce lien : https://klee.github.io/tutorials/testing-regex/
il plus recommandé de préférence d’utiliser les opérateur | et & plutôt que || et && dans la construction des contraints pour klee_assume
car dans le deuxième cas, klee traduit l’ensemble en blocs de branches { if else ..} avant d’assumer ces contrainte ce qui les rend plus simple, mais toutefois moins

Du coup la deuxième essai d’utilisation de klee est la suivante :
j’ai insérer des assert pour envoyer un signal en cas d’erreur dans les fichiers suivants :
dico.c

```
#ifdef KLEE
#include <klee/klee.h>
#include <assert.h>
int deco (int argc, char **argv)
{
	unsigned int* line = (unsigned int*) malloc(sizeof(int));
	klee_make_symbolic(line, sizeof(int), "line");
	unsigned int* colonne = (unsigned int*) malloc(sizeof(int));
	klee_make_symbolic(colonne, sizeof(int), "colonne");
	dico* dictionary = (dico*) malloc(sizeof(dico));
	dictionary->mot=NULL;
	dictionary->next=NULL;
	unsigned int maxSizeDico;
	klee_make_symbolic(&maxSizeDico, sizeof(int), "maxSizeDico");
	klee_assume(maxSizeDico < 10);// la limite reste à définir
	klee_assume(maxSizeDico > 0);
	for (int i=0; i<maxSizeDico ; i++){
		dictionary = addToDico(dictionary, argv[1],line,colonne);
		if (dictionary->mot== NULL) {
			assert(0);
		}
	}

	dico* tempDico = (dico*) malloc(sizeof(dico));
	tempDico = dictionary;
	mot_t* w1=tempDico->mot;
	mot_t* w2=NULL;
	tempDico = tempDico->next;

  while(tempDico != NULL) {
    w2=tempDico->mot;
    if (compareWord( w1,  w2) > -1){
```

```
            assert(0);
        }
        w1=w2;
        tempDico = tempDico->next;
    }
    free(tempDico);
            free(line);
            free(colonne);
            free(dictionary);
            return 0;
}
```

Le mot est representé par le premier argument argv[1]
j'ai préférer de symbolisé toutes les variable essentiel pour un test plus ou moins complet :
+le mot, ligne et colonne
+nombre de mot insérer dans le dictionaire
et j'ai rajouter un morceau de code pour traiter la vérification du bon d'ordre d'insertion dans le dictionnaire.

words.c

```
int compareWord(mot_t* w1, mot_t* w2) {
  if ((w1 == NULL)&&(w2 == NULL)){
            #ifdef KLEE
            assert(0);
            #endif
            return 0;
  }
  if (w1 == NULL) {
            #ifdef KLEE
            assert(0);
            #endif
    return -1;
  } else if (w2 == NULL) {
            #ifdef KLEE
            assert(0);
            #endif
    return 1;
  } else {
    char* word1 = maillonToString(w1->tete_mot);
    char* word2 = maillonToString(w2->tete_mot);
    int minSize = (strlen(word1)<strlen(word2))?strlen(word1):strlen(word2);
    int i = 0;
    int pos = 0;
    while(i<minSize && pos == 0) {
      pos = (word1[i]<word2[i])?-1:(word1[i]>word2[i])?1:0;
      i++;
    }
    return (pos == 0 && strlen(word1) < strlen(word2))?-1:(pos == 0 && strlen(word1) >
strlen(word2))?1:pos;
```

```
  }
}
```

maillons.c :

```
#include "maillons.h"

void setCharnum(maillon_t* link, int k, char c) {
  if (link == NULL) {
  #ifdef KLEE
  assert(0);
  #endif
    printf("setCharnum : link null\n");
  } else {
    link->elem = (link->elem & ~getMask(k)) + ((unsigned int)charToNum(c) << (5*(5-k)));
  }
}

char getCharnum(maillon_t* link, int k) {
  if (link == NULL) {
    printf("getCharnum : link null\n");
    #ifdef KLEE
    assert(0);
    #endif
    return '#';
  } else {
    return numToChar((link->elem & getMask(k)) >> (5*(5-k)));
  }
}

int isAvailable(char c) {
  if (c < 'a' || c > 'z') {
    return 0;
  } else if (c =='\0') {
    return 1;
  }
  #ifdef KLEE
  assert(0);
  #endif
  return 2;
}

int getSizeMaillon(maillon_t* link) {
  if (link == NULL) {
    return 0;
```

```
   #ifdef KLEE
   assert(0);
   #endif
  } else {
   int i = 0,res = 0;
   maillon_t* useLink = link;
   while(useLink != NULL) {
    for(i=0;i<=5;i++) {
        if  (isAvailable(getCharnum(useLink,i)) == 0) {break;}
        res++;
    }
    useLink = useLink->next;
   }
        return res;
  }
}

int charToNum(char c) {
  return c - 'a' + 1;
}

char numToChar(int i) {
  return (i + 'a' - 1);
}

char* maillonToString(maillon_t* link) {
  if (link == NULL) {
   #ifdef KLEE
   assert(0);
   #endif
   return NULL;

  } else {
   char* word = (char*) malloc(sizeof(char)*getSizeMaillon(link)+1);
   maillon_t* useLink = link;
   int index = 0, i = 0;
   while (useLink != NULL) {
    for(i=0;i<=5;i++) {
        if (isAvailable(getCharnum(useLink,i)) == 0) {break;}
        else {
          word[index] = getCharnum(useLink,i);
          index++;
        }
    }
    useLink = useLink->next;
   }
   word[index] = '\0';//word[index+1] = '\0';
   free(useLink);
```

```
    return word;
  }
}
maillon_t* stringToMaillon(char* word) {
  if (word == NULL) {
  #ifdef KLEE
  assert(0);
  #endif
    return NULL;
  } else {
    maillon_t* useLink = (maillon_t*) malloc(sizeof(maillon_t));
    useLink->elem = 0;
    useLink->next = NULL;
    maillon_t* savLink = useLink;
    int i;
    for(i=0;i<strlen(word);i++) {
      if ((i % 6) == 0 && useLink->elem != 0) {
            maillon_t* newLink = (maillon_t*) malloc(sizeof(maillon_t));
            newLink->elem = 0;
            newLink->next = NULL;
            useLink->next = newLink;
            useLink = newLink;
      }
      setCharnum(useLink,(i%6),word[i]);
    }
    return savLink;
  }
}

unsigned int getMask(int k) {
  switch(k) {
  case 0:
    return 0x3E000000;
    break;
  case 1:
    return 0x01F00000;
    break;
  case 2:
    return 0x000F8000;
    break;
  case 3:
    return 0x00007C00;
    break;
  case 4:
    return 0x000003E0;
    break;
  case 5:
    return 0x0000001F;
```

```
    break;
  }
  #ifdef KLEE
  assert(0);
  #endif
  return 0x00000000;
}
```

La compilation avec le makefile éxécute les commandes suivantes :

```
$ make klee=1
wllvm -Wall -g  -c src/dico.c -I/home/m/monniaud/packages/klee/2017-09-26_c7a1f9d/include -I
include -DKLEE -o objs/dico.o
wllvm -Wall -g  -c src/words.c -I include -o objs/words.o
wllvm -Wall -g  -c src/maillons.c -I include -o objs/maillons.o
wllvm -Wall -g  -c src/display.c -I include -o objs/display.o
wllvm -Wall -g  -c src/AllTests.c -I include -o objs/AllTests.o
wllvm -Wall -g  -c src/CuTest.c -I include -o objs/CuTest.o
wllvm -Wall -g  -c src/SuiteDeTests.c -I include -o objs/SuiteDeTests.o
wllvm -L/home/m/monniaud/packages/klee/2017-09-26_c7a1f9d/lib -lkleeRuntest -o objs/projet2
objs/main.o objs/dico.o objs/words.o objs/maillons.o objs/display.o objs/AllTests.o objs/CuTest.o
objs/SuiteDeTests.o
extract-bc objs/projet2
```

Et l’exécution avec klee se fait par :

```
klee --optimize --libc=uclibc --posix-runtime objs/projet2.bc --sym-arg 25
```

Tel que :
--optimize : permet d’ignorer le code mort
--libc=uclibc --posix-runtime : en cas d’utilisation de librairie standard
--sym-arg N : permet de symboliser la chaine de caractère argv[1] donné en d’entrée de taille max N.
l’exécution prend beaucoup de temps
voici quelque fichiers tests résultants à titre d’exemple :

```
$ ktest-tool objs/klee-out-1/test00005*.ktest
ktest file : 'objs/klee-out-1/test000050.ktest'
args       : ['objs/projet2.bc', '--sym-arg', '25']
num objects: 5
object    0: name: 'arg0'
object    0: size: 26
object    0: data: '\x1b\x06\x01\x01\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00'
object    1: name: 'model_version'
object    1: size: 4
object    1: data: '\x01\x00\x00\x00'
object    2: name: 'line'
object    2: size: 4
object    2: data: '\x00\x00\x00\x00'
object    3: name: 'colonne'
```

```
object    3: size: 4
object    3: data: '\x00\x00\x00\x00'
object    4: name: 'maxSizeDico'
object    4: size: 4
object    4: data: '\x01\x00\x00\x00'
()
ktest file : 'objs/klee-out-1/test000051.ktest'
args       : ['objs/projet2.bc', '--sym-arg', '25']
num objects: 5
object    0: name: 'arg0'
object    0: size: 26
object    0: data: '\x06\x01\x01\x01\x01\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00'
object    1: name: 'model_version'
object    1: size: 4
object    1: data: '\x01\x00\x00\x00'
object    2: name: 'line'
object    2: size: 4
object    2: data: '\x00\x00\x00\x00'
object    3: name: 'colonne'
object    3: size: 4
object    3: data: '\x00\x00\x00\x00'
object    4: name: 'maxSizeDico'
object    4: size: 4
object    4: data: '\x01\x00\x00\x00'
()
ktest file : 'objs/klee-out-1/test000052.ktest'
args       : ['objs/projet2.bc', '--sym-arg', '25']
num objects: 5
object    0: name: 'arg0'
object    0: size: 26
object    0: data: '\x1b\x06\x01\x01\x01\x01\x1c\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00'
object    1: name: 'model_version'
object    1: size: 4
object    1: data: '\x01\x00\x00\x00'
object    2: name: 'line'
object    2: size: 4
object    2: data: '\x00\x00\x00\x00'
object    3: name: 'colonne'
object    3: size: 4
object    3: data: '\x00\x00\x00\x00'
object    4: name: 'maxSizeDico'
object    4: size: 4
object    4: data: '\x01\x00\x00\x00'
()
ktest file : 'objs/klee-out-1/test000053.ktest'
args       : ['objs/projet2.bc', '--sym-arg', '25']
```

```
num objects: 5
object    0: name: 'arg0'
object    0: size: 26
object    0: data: '\x1c\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00'
object    1: name: 'model_version'
object    1: size: 4
object    1: data: '\x01\x00\x00\x00'
object    2: name: 'line'
object    2: size: 4
object    2: data: '\x00\x00\x00\x00'
object    3: name: 'colonne'
object    3: size: 4
object    3: data: '\x00\x00\x00\x00'
object    4: name: 'maxSizeDico'
object    4: size: 4
object    4: data: '\x08\x00\x00\x00'
()
ktest file : 'objs/klee-out-1/test000054.ktest'
args       : ['objs/projet2.bc', '--sym-arg', '25']
num objects: 5
object    0: name: 'arg0'
object    0: size: 26
object    0: data: '\x03(\x07\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00'
object    1: name: 'model_version'
object    1: size: 4
object    1: data: '\x01\x00\x00\x00'
object    2: name: 'line'
object    2: size: 4
object    2: data: '\x00\x00\x00\x00'
object    3: name: 'colonne'
object    3: size: 4
object    3: data: '\x00\x00\x00\x00'
object    4: name: 'maxSizeDico'
object    4: size: 4
object    4: data: '\x01\x00\x00\x00'
()
ktest file : 'objs/klee-out-1/test000055.ktest'
args       : ['objs/projet2.bc', '--sym-arg', '25']
num objects: 5
object    0: name: 'arg0'
object    0: size: 26
object    0: data: '\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00'
object    1: name: 'model_version'
object    1: size: 4
object    1: data: '\x01\x00\x00\x00'
```

```
object    2: name: 'line'
object    2: size: 4
object    2: data: '\x00\x00\x00\x00'
object    3: name: 'colonne'
object    3: size: 4
object    3: data: '\x00\x00\x00\x00'
object    4: name: 'maxSizeDico'
object    4: size: 4
object    4: data: '\x07\x00\x00\x00'
()
ktest file : 'objs/klee-out-1/test000056.ktest'
args       : ['objs/projet2.bc', '--sym-arg', '25']
num objects: 5
object    0: name: 'arg0'
object    0: size: 26
object    0: data: '\x1c\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00'
object    1: name: 'model_version'
object    1: size: 4
object    1: data: '\x01\x00\x00\x00'
object    2: name: 'line'
object    2: size: 4
object    2: data: '\x00\x00\x00\x00'
object    3: name: 'colonne'
object    3: size: 4
object    3: data: '\x00\x00\x00\x00'
object    4: name: 'maxSizeDico'
object    4: size: 4
object    4: data: '\t\x00\x00\x00'
()
ktest file : 'objs/klee-out-1/test000057.ktest'
args       : ['objs/projet2.bc', '--sym-arg', '25']
num objects: 5
object    0: name: 'arg0'
object    0: size: 26
object    0: data: '\x89S\x07\x01\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00'
object    1: name: 'model_version'
object    1: size: 4
object    1: data: '\x01\x00\x00\x00'
object    2: name: 'line'
object    2: size: 4
object    2: data: '\x00\x00\x00\x00'
object    3: name: 'colonne'
object    3: size: 4
object    3: data: '\x00\x00\x00\x00'
object    4: name: 'maxSizeDico'
object    4: size: 4
```

```
object    4: data: '\x01\x00\x00\x00'
()
ktest file : 'objs/klee-out-1/test000058.ktest'
args       : ['objs/projet2.bc', '--sym-arg', '25']
num objects: 5
object    0: name: 'arg0'
object    0: size: 26
object    0: data: '\x1b\x06\x01\x01\x01\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00'
object    1: name: 'model_version'
object    1: size: 4
object    1: data: '\x01\x00\x00\x00'
object    2: name: 'line'
object    2: size: 4
object    2: data: '\x00\x00\x00\x00'
object    3: name: 'colonne'
object    3: size: 4
object    3: data: '\x00\x00\x00\x00'
object    4: name: 'maxSizeDico'
object    4: size: 4
object    4: data: '\x01\x00\x00\x00'
()
ktest file : 'objs/klee-out-1/test000059.ktest'
args       : ['objs/projet2.bc', '--sym-arg', '25']
num objects: 5
object    0: name: 'arg0'
object    0: size: 26
object    0: data: '\x89S\x07\x01\x01\x01\x1c\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00'
object    1: name: 'model_version'
object    1: size: 4
object    1: data: '\x01\x00\x00\x00'
object    2: name: 'line'
object    2: size: 4
object    2: data: '\x00\x00\x00\x00'
object    3: name: 'colonne'
object    3: size: 4
object    3: data: '\x00\x00\x00\x00'
object    4: name: 'maxSizeDico'
object    4: size: 4
object    4: data: '\x01\x00\x00\x00'
```

```
$ ll objs/klee-out-0/
Display all 60547 possibilities? (y or n)
```

```
$ ll objs/klee-out-0/*.err
ls: impossible d'accéder à objs/klee-out-0/*.err: Aucun fichier ou dossier de ce type
```

Sources :
https://klee.github.io/tutorials/testing-regex/
http://feliam.wordpress.com/2010/10/07/the-symbolic-maze/
https://gitlab.com/Manouchehri/Matryoshka-Stage-2/blob/master/stage2.md
https://doar-e.github.io/blog/2015/08/18/keygenning-with-klee/

Conclusion :
je l'ai laissé tourné pendant des heures sauf qu'on remarque que il n'existe aucun fichier .err dans le répertoire crée par klee.
Du coup, et en se basant sur le tutoriel décrit en suivant ce lien :