Functionnality of Forester

Cédric Vernou
Programmeur émérite

18 juillet 2013

Table des matières

1 Introduction

In my school project, I have worked on Forester. Presentely, Forester don't have documentation and I have exchanged a lot of email with authors of Forester. This document is a synthesis of this explanation and my experimentations.

Forester is an experimental tool for checking manipulation of dynamic data structures using *Forest Automata*. The tool can be loaded into *GCC* as a *plug-in*. This way you can easily analyse C code sources, using the existing build system, without manually preprocessing them first. However, the analysis itself is not yet mature enough to be able to verify ordinary programs.

2 Use Forester

2.1 To use

Forester is plugin GCC, you can use this command to load plugin in GCC and check code source :

gcc -fplugin=libfa.so

But Forester is provided with some script for ease of use. In the first place, you need config some variable in console with this script "./fa_build/register-paths.sh". You can use this command :

. ./fa_build/register-paths.sh

This script add path in you terminal to other script to execute easy Forester. Now, you can use Forester scripts to execute GCC and to load Forester. The main script is :

fagcc main.c

2.2 Some remarks

This scrips are really easy to use, but they have default. You can check one file at at time.

Other remark who are problemtatic with the precedent, the only file in parameter have to containt the method main. Else Forester return this error :

```
error: main() not found at global scope
```

A method to check your files, it is use a other file forester.c and include all files. In forester.c, write main method where you call all methods you want test.

```
        #include "list.c"

        int main()
        {
                Node * list = initializeList(1);
                finalizeList(list);
                return 0;
        }
```

3 Forester fonctionnalities

3.1 Type primitif

The manage of type primitif in Forester is not again implemented. If you use pointer on primitif type, Forester return this error :

```
        int * integer = malloc(sizeof(int));
```

Forester return this resultat :

```
error: BoxMan::getTypeInfo(): type for int not found!
```

The reason is primitif types have different representation in GCC. Them, Forester can not recognize its. This fonctionnality is comming soon.

But presentely, you can get around this problem with this solution :

```
        struct T { int n; }* p_int;
        p_int = malloc(sizeof(*p_int));
        p_int->n = 10;
        free(p_int);
```

3.2 To use no allocated structure

Before use dynamic structure, he need to be allocated in memory. Forester check if all stucture are corectely allocated before she is used. This example show dynamic stucture who are used without allocation :

```
        struct T { int n; }* pmi;
        pmi->n = 10;
```

In this case, Forester return this error :

```
unallocated\_structure.c:7:9: note: #2317:pmi->n = (int)10
unallocated\_structure.c:7:9: error: dereferenced value is not a valid
        reference [(undef)]
```

3.3 Dynamic structure not desallocated

Forester can find when a dynamic structure, who is correctely allocated, is not desallocated. In this example, pmi is allocated but this structure is not allocated and the reference is lost after function's end :

```
        struct T { int n; }* pmi;
        pmi = malloc(sizeof(*pmi));
        pmi->n = 10;
        return;
```

In this case, Forester return this error :

```
error: garbage detected
```

3.4 Reference to dynamic structure no catch in return of function

Sometime, a function allocate a dynamic structure and return the reference. In this case, it is important catch this reference to desallocated this dynamic structure. Forester check if all reference return by function is correctely catch.

```c
struct T { int n; };
typedef struct T T;

T * foe()
{
        return malloc(sizeof(T));
}

int main()
{
        foe();
        return 0;
}
```

In this case, Forester return this error :

```
error: assertion failed
```

This fonctionnality is user-friendly to avoid memory leaks. But it is posible function return reference to structure who are not allocated in this function. Forester return same error this return is not catch.

```c
struct T { int n; };
typedef struct T T;

T * foe(T * t)
{
        return t;
}

int main()
{
        T t;
        foe(&t);
        return 0;
}
```

Glossary

Forest Automata Tuples of tree automata which accept trees whose leaves can refer to the roots of all trees accepted by these automata. 2

Références