Frama-C installation and Overview

Stance Training Session – Course 1

Virgile Prevosto

March 28th, 2013

Outline

Installation of Frama-C

Frama-C Overview
- Brief history
- Quick Presentation

Frama-C Kernel

Plug-ins

For the training session

Installation

- If you don't have a virtualizer, use the appropriate `VMWare Player` provided on the USB stick
- Import the virtual machine on `VMWare` or `VirtualBox`.
- Launch the virtual machine.

Spec of the virtual machine

- based on Xubuntu 12.10
- Frama-C Oxygen, Why 2.31, Why3 0.8
- Alt-ergo 0.94, Z3 3.2, Simplify, Coq
- Verifast

On Linux

- On Debian, Ubuntu, Fedora, Gentoo, OpenSuse, Linux Mint, …
- Compile from sources using OCaml package managers:
  - Godi (`http://godi.camlcity.org/godi/index.html`)
  - Opam (`http://opam.ocamlpro.com/`)

On Windows

- Godi
- Wodi (`http://wodi.forge.ocamlcore.org/`)

Installed files

Executables

- frama-c: Console-based interface
- frama-c-gui: Graphical User Interface

Others

- FRAMAC_PLUGINS: location of plug-ins
- FRAMAC_SHARE: various configuration files
- FRAMAC_SHARE/libc: standard headers

Documentation

Manuals

- `http://frama-c.com/support.html`
- In directory `$(frama-c -print-share-path)/manuals`

Support

- `frama-c-discuss@gforge.inria.fr`
- tag `frama-c` on `http://stackoverflow.com`

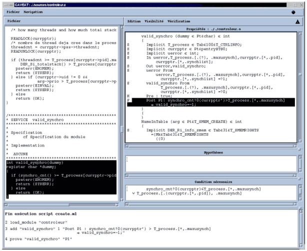
Inline summary

- `frama-c -help`
- `frama-c -kernel-help`
- `frama-c -⋆-path`
- `frama-c -`*`plugin`*`-help`

Caveat Verifier

```
Pre     params:    a>=0 && b>0;
Term 1  variant:   a-b;
Inv  1  invariant: a+b*q=a';
Post    remainder: a-b*div < b;
```

```
int div(int a, int b) {
  int q;
  q=0;
  while (a>=b) {
    a=a-b;
    q=q+1;
  }
  return q;
}
```



```
  /* how many threads and how much total stack

  READLOCK(currpptr);
  /* nombre de thread deja cree dans le proces
  threadcnt = currpptr->pss->threadcnt;
  READUNLOCK(currpptr);

  if (threadcnt >= T_process[currpptr->pid].ma
      DER_Si_totalstack() > T_process[currpptr
    pseterr(ENOMEM);
    return (SYSERR);
  } else if (currpptr->uid != 0 &&
             arg->prio > T_process[currpptr->p
    pseterr(EINVAL);
    return (SYSERR);
  } else
    return (OK);
}


/*********************************************
* SERVICE  valid_synchro
**********************************************
*
* Specification
*       cf Specification du module
*
* Implementation
*
*  AUCUNE
**********************************************
int valid_synchro(dummy)
register char *dummy;
{
  if (synchro_cnt() >= T_process[currpptr->pid
    pseterr(ENOMEM);
    return (SYSERR);
  } else
    return (OK);
}
```

Propriétés : ./.controleur.s

```
      valid_synchro (dummy ∈ Pt&char) ∈ int
      {
S       Implicit T_process ∈ Tab&100&T_CTRLINFO;
S       Implicit currpptr ∈ Pt&pentry@T@0;
S       Implicit uerror ∈ int;
S       In uerror,T_process.[.(?), .maxusynch],currpptr.[*,.pid],
            currpptr.[*,.synchlist];
S       Out uerror,valid_synchro;
S       uerror From
            uerror,T_process.[.(?), .maxusynch],currpptr.[*,.pid],
              currpptr.[*,.synchlist] =?1;
S       valid_synchro From
              T_process.[.(?), .maxusynch],currpptr.[*,.pid],
              currpptr.[*,.synchlist] =?0;
H       Pre : true;
P       Post P1: synchro_cnt?0(currpptr')>T_process.[*,.maxusynch]
          ⇒ valid_synchro=-1;
      }
      {
      }
      NameInTable (arg ∈ Pt&T_SMEM_CREATE) ∈ int
      {
S       Implicit DER_Ri_info_smem ∈ Tab&31&T_SMEMRIGHTS
          =(Mk&Tab&31&T_SMEMRIGHTS
            ((0)
```

Hypothèses

Condition nécessaire

```
      synchro_cnt?0(currpptr)≤T_process.[*,.maxusynch]
  ∨ T_process.[.(currpptr.[*,.pid]), .maxusynch]
```

Fin exécution script create.ml

```
2 load_module "controleur"
3 add "valid_synchro" 1 "Post P1 : synchro_cnt?0(currpptr') > T_process.[*,.maxusynch]
                         ⇒ valid_synchro=-1;"
4 prove "valid_synchro" "P1"
```

```
remainder: Proved
           (Simplify,Z3,Alt-Ergo)
```

Caveat @ Airbus

- Experiments with Caveat since 1998
- Used in some critical developments (qualified for DO-178B level A on this code)
- Replaces unit tests by formal proofs
- J. Souyris & al. *Formal Verification of Avionics Software Products*, FM 2009, vol. 5850 LNCS

Caveat and Frama-C

Reinvesting Caveat pro's

- Formal language designed for code specifications
- Hoare's logic, weakest preconditions

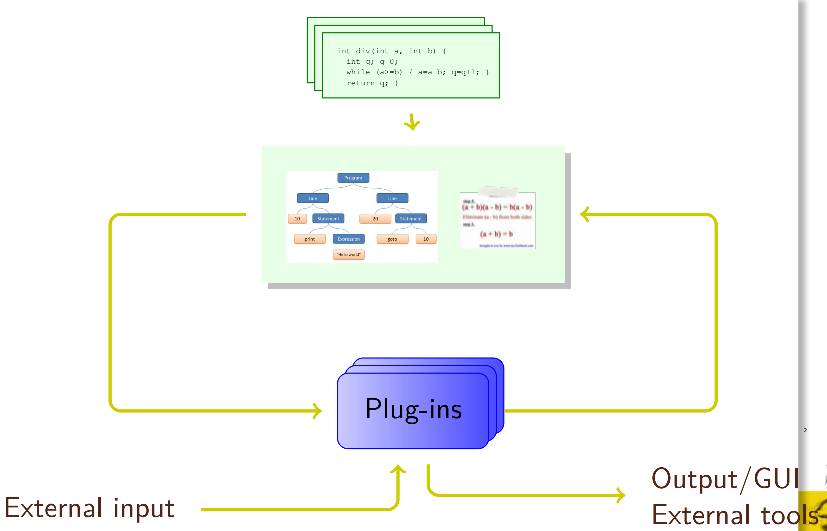
Improving scope of application

- Low-level C-code features (complex aliases, casts)
- Other semantic analysis (static analysis by abstract interpretation)

Frama-C at a glance

- `http://frama-c.com/`
- Developed at CEA LIST and INRIA Saclay (Proval team, now Toccata).
- Released under LGPL license.
- Kernel based on CIL (Necula et al. – Berkeley).
- ACSL annotation language.
- Extensible platform
  - Collaboration of analysis over same code
  - Inter plug-in communication through ACSL formulas.
  - Adding specialized plug-in is easy

Frama-C platform



```
int div(int a, int b) {
  int q; q=0;
  while (a>=b) { a=a-b; q=q+1; }
  return q; }
```

Plug-ins

External input

Output/GUI
External tools

CIL and ACSL

Abstract syntax trees

- Parsing and type-checking
- Normalization
- Code transformation (visitor)
- Control-flow graph and data-flow analysis

Projects and Journalization

Managing the state of the analyzer

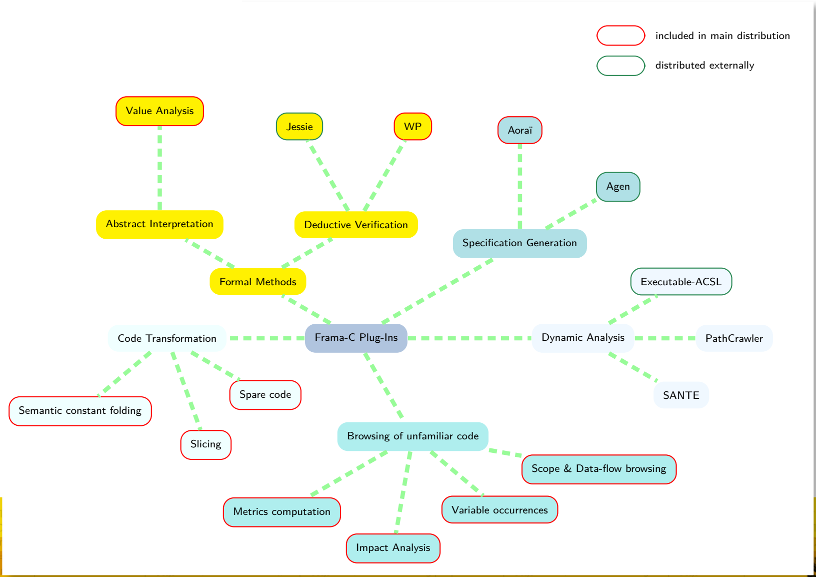
- Encompasses Frama-C's internal state
- Two projects are independent from each other
- Persistence (type-safe load/save)
- Replay features through journalized script.

Example

```
module Done =
  Computation.Ref
    (struct ... end)(struct ... end)

let project =
  File.create_project_from_visitor "transf"
    (fun p -> new my_code_transformer p)
```

Main plugins



🟥 included in main distribution

🟩 distributed externally

Frama-C Plug-Ins

Formal Methods

Abstract Interpretation

Value Analysis

Deductive Verification

Jessie

WP

Specification Generation

Aoraï

Agen

Dynamic Analysis

Executable-ACSL

PathCrawler

SANTE

Code Transformation

Semantic constant folding

Slicing

Spare code

Browsing of unfamiliar code

Scope & Data-flow browsing

Variable occurrences

Metrics computation

Impact Analysis

External plugins

- Taster (coding rules, Atos/Airbus, Delmas &al., ERTS 2010)
- Dassault's internal plug-ins (Pariente & Ledinot, FoVeOOs 2010)
- Fan-C (flow dependencies, Atos/Airbus, Duprat &al., ERTS 2012)
- Various academic experiments (mostly security-related)

Plug-ins

Registering a new plug-in

- Inform the kernel of the plug-in
- Register plug-in state in project mechanism
- Register exposed functions in the dynamic mechanism
- Register entry point in the kernel

Example

```
module P = Plugin.Register(struct ... end)
module Enabled = P.False(struct ... end)
let print () = P.result "Hello world";;
Db.Extend.main
  (fun () -> if Enabled.get() then print ());;
```